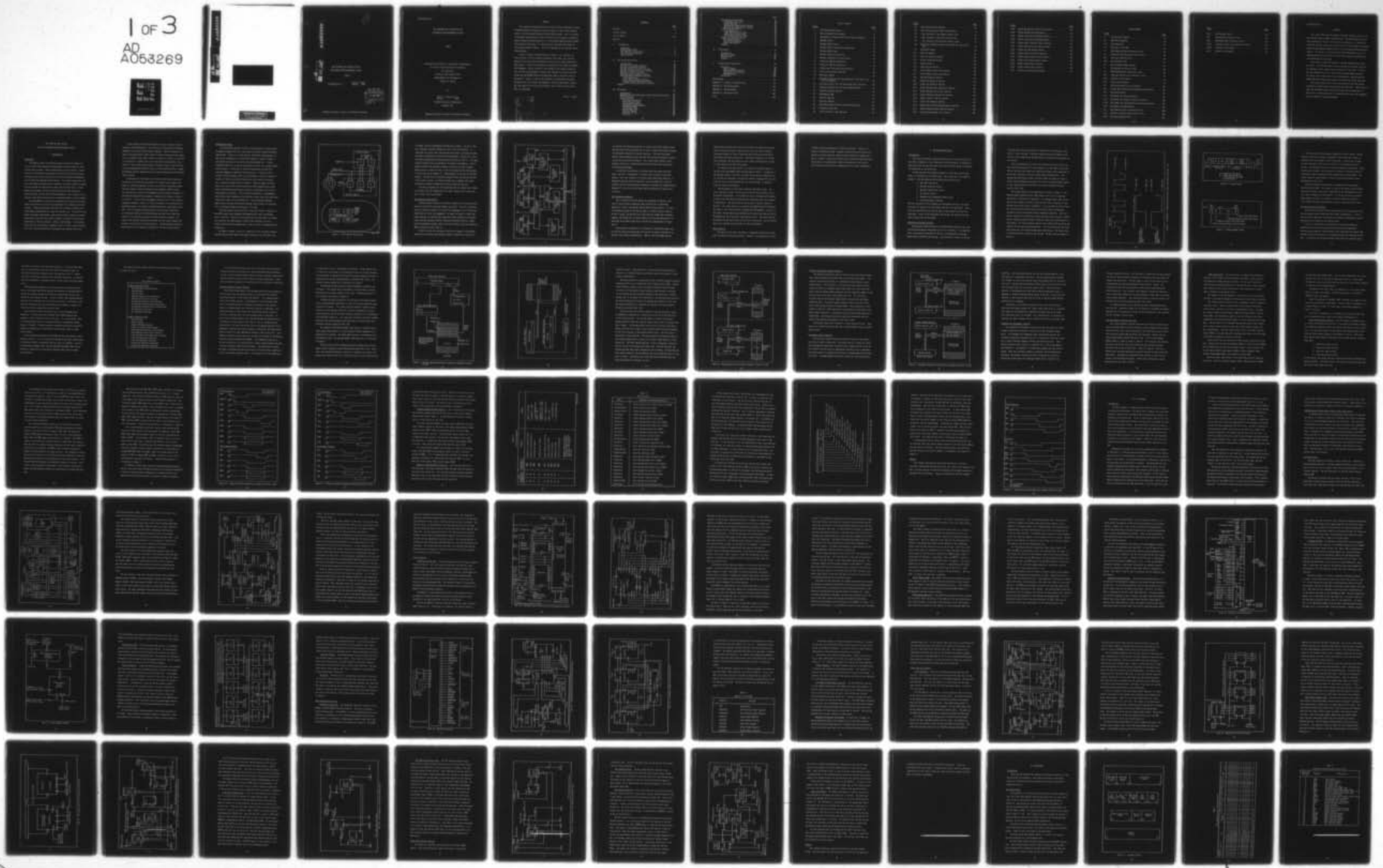
AD-A053 269   AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO   SCH--ETC   F/G 9/2
BUS INTERFACE UNIT DESIGN FOR THE DISTRIBUTED PROCESSOR/MEMORY --ETC(U)
DEC 77   L T KONNO

UNCLASSIFIED   AFIT/GE/EE/77-24   NL

1 OF 3
AD
A053269



BUS INTERFACE UNIT DESIGN FOR THE DISTRIBUTED PROCESSOR/MEMORY SYSTEM

THESIS

Leslie T. Konno
Captain USAF

AFIT/GE/EE/77-24

Approved for public release; distribution unlimited.

AFIT/GE/EE/77-24

BUS INTERFACE UNIT DESIGN FOR THE DISTRIBUTED PROCESSOR/MEMORY SYSTEM

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the degree of
Master of Science

by

Leslie T. Konno, B.S.E.E.
Captain USAF

Graduate Electrical Engineering

December 1977

PREFACE

This study was sponsored by the Air Force Avionics Laboratory, System Technology Branch, Software and Processor Group, as part of the development of the Distributed/Processor Memory (DP/M) system. This is a system to integrate the avionics on board an aircraft with a group of processor/memory elements (microprocessors) in a distributed (decentralized) network. The purpose of the study is to design the Bus Interface Unit (BIU) for the processor/memory element. The BIU is designed as an interrupt driven microprocessor.

I want to express my appreciation to Captain J. B. Peterson, my thesis advisor, for his guidance throughout this study, and to Deiter Schiller from the Avionics Laboratory for his help in understanding the DP/M system. I want to also express my appreciation to Flight Lieutenant Ejaz Muhammad. He was involved in the design of the central processor unit for the processor/memory element; and together we learned the DP/M system and the Am2900 Bipolar Microprocessor Family, which was used in the design. Finally, I want to pay tribute to the patience and understanding of my wife, JoAnn, and daughters, Christine and Cheryl, during the time spent on this study and during my tour at the Air Force Institute of Technology.

Leslie T. Konno

Contents

Page

Page

List of Figures

Figure Page

List of Tables

ABSTRACT

This report describes the design of the Bus Interface Unit for the Distributed Processor/Memory (DP/M) System. The DP/M System, which is being developed by the Air Force Avionics Laboratory, is a concept to integrate avionics on board an aircraft by utilizing a number of programmable processor/memory elements (PE's) in a distributed (decentralized) network. All the PE's in the system are interconnected by a pair of redundant global buses and PE's in an affinity group are additionally interconnected by a local bus.

This effort involves the design of the Bus Interface Unit (BIU) of the PE. The BIU interfaces the parallel-data PE processor with the two redundant serial global buses and the serial local bus. The BIU has been designed as an interrupt driven microprogrammed processor. The design uses a bipolar bit-slice microprocessor, specifically the Am2900 Bipolar Microprocessor Family, for emulating the BIU functions.

This report starts with a brief description of the DP/M System, followed by a detailed description of the BIU functions. Then the hardware, the microword format, and the microroutines are described. A discussion on the design effort is presented at the end, and recommendations are made for system improvement.

BUS INTERFACE UNIT DESIGN

FOR THE DISTRIBUTED PROCESSOR/MEMORY SYSTEM

I Introduction

Background

The complex avionic and control systems required for modern Air Force aircraft have required the development of new concepts for integrating these systems. Modern sophisticated aircraft require a large amount of in-flight data processing for effective mission performance. Previously, most avionic systems on an aircraft operated independently of each other and employed analog devices. As aircraft became more complex, the onboard systems became integrated, and these systems increased aircraft weight and required more space and electrical power. Better methods were needed to process and to provide the data required to achieve optimal aircraft and mission performance. This resulted in new concepts using digital processors for integrating these aircraft systems.

The development of Large Scale Integrated (LSI) circuitry and small digital processors has led to many new methods for integrating aircraft avionics. These new processors provide fast, accurate, reliable, and light weight systems which are ideal for avionics use. To meet the size, weight, and power requirements, many current avionic systems are highly centralized, relying on one central computer to process and distribute information to various actuators, sensors, and displays. This concept relies heavily on the central computer, and if it fails or the interface to it fails the aircraft mission performance may degrade drastically.

If the system could be decentralized by using a group of smaller processors (microprocessors), the resulting system would not be highly dependent on one specialized central computer. Each microprocessor or group of microprocessors would receive raw data from one aircraft sensor (e.g., airspeed, loran, radar, doppler), process it, format it, and make it available on a "global" bus to all the other microprocessors that need it. This system concept should be highly reliable, since the loss of one microprocessor would not result in total system failure. System performance would be degraded but still highly effective and the mission could continue.

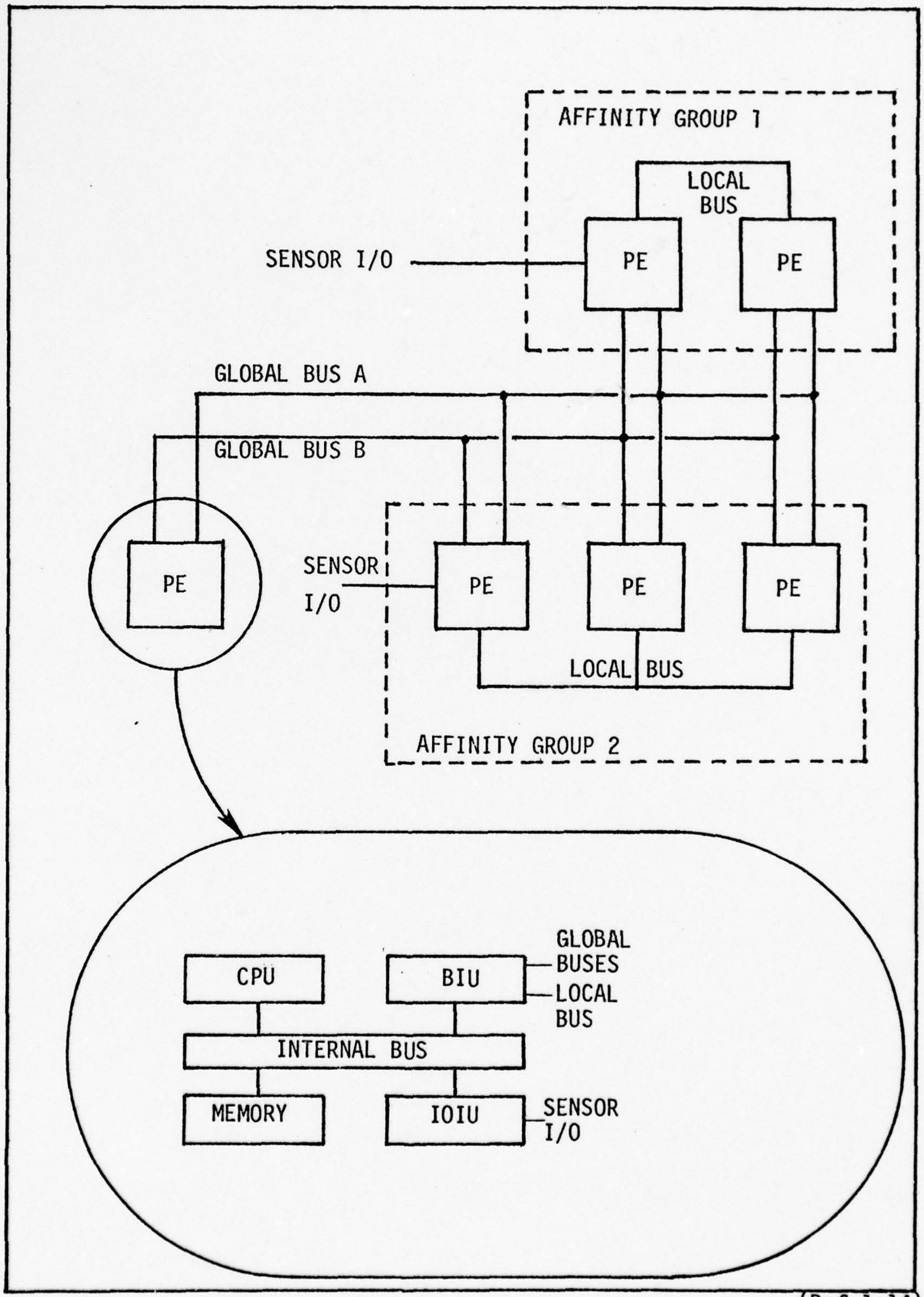
To overcome the limitations of the centralized digital processor system and still meet the requirements for reduced size, weight, and power for airborne equipment, the Air Force Avionics Laboratory formalized the concept of the Distributed Processor/Memory (DP/M) System. This system uses a number of programmable processor/memory elements in a distributed (decentralized) system to integrate the avionics on board an aircraft. As part of the development program, the Air Force Avionics Laboratory awarded a contract to Texas Instruments (TI) Incorporated to determine whether this type of decentralization was functionally feasible. The TI program consisted primarily of four major objectives: (1) functional design of the DP/M hardware and high level simulation analysis, (2) functional design of the DP/M executive software, (3) study of the possible construction methods for the system hardware and software, and (4) fault tolerance analysis. The DP/M System concept as conceived by the TI program is presented in the next section (Ref 1).

DP/M System Concept

The DP/M System concept utilizes a varying number of simple processor/memory elements (PE) for a wide range of avionic system processing applications. These PE's can be used as stand alone uniprocessors, or they can be configured in a distributed network as shown in Figure 1. The network consists of two levels of busing between PE's. A dual-redundant global bus is used to interconnect each PE in a system communication network. A local bus is used to interconnect multiple PE's clustered together to perform a given function. A cluster of PE's, called an Affinity Group, would be required when a single PE could not process all the data available from one aircraft sensor or actuator (e.g., hydraulic system, control surface). Both the global and local buses use a serial time-division-multiplex (TDM) communication system with a distributed round-robin bus protocol scheme where each PE has a predefined bus allocation. This allocation mechanism is programmable, which allows super-commutation (multiple bus access time slots within one round-robin cycle) of bus transmissions. Each transmitted message begins with a message identification header, which allows all recipient PE's to determine whether the message is for them.

Since the global busing facility is the only "central" resource of the DP/M system, dual-redundant TDM communication lines interconnect each of the PE's. Every PE on the global bus can participate in primary bus transfers and simultaneously listen on the secondary bus for a "switch to back-up bus" command when a fault or error is detected on the primary bus.

As shown in Figure 1, each PE is made up of four functional modules: the Bus Interface Unit (BIU), the Central Processor Unit (PE CPU), the



(Ref 1:14)

Figure 1. DP/ M System Architecture

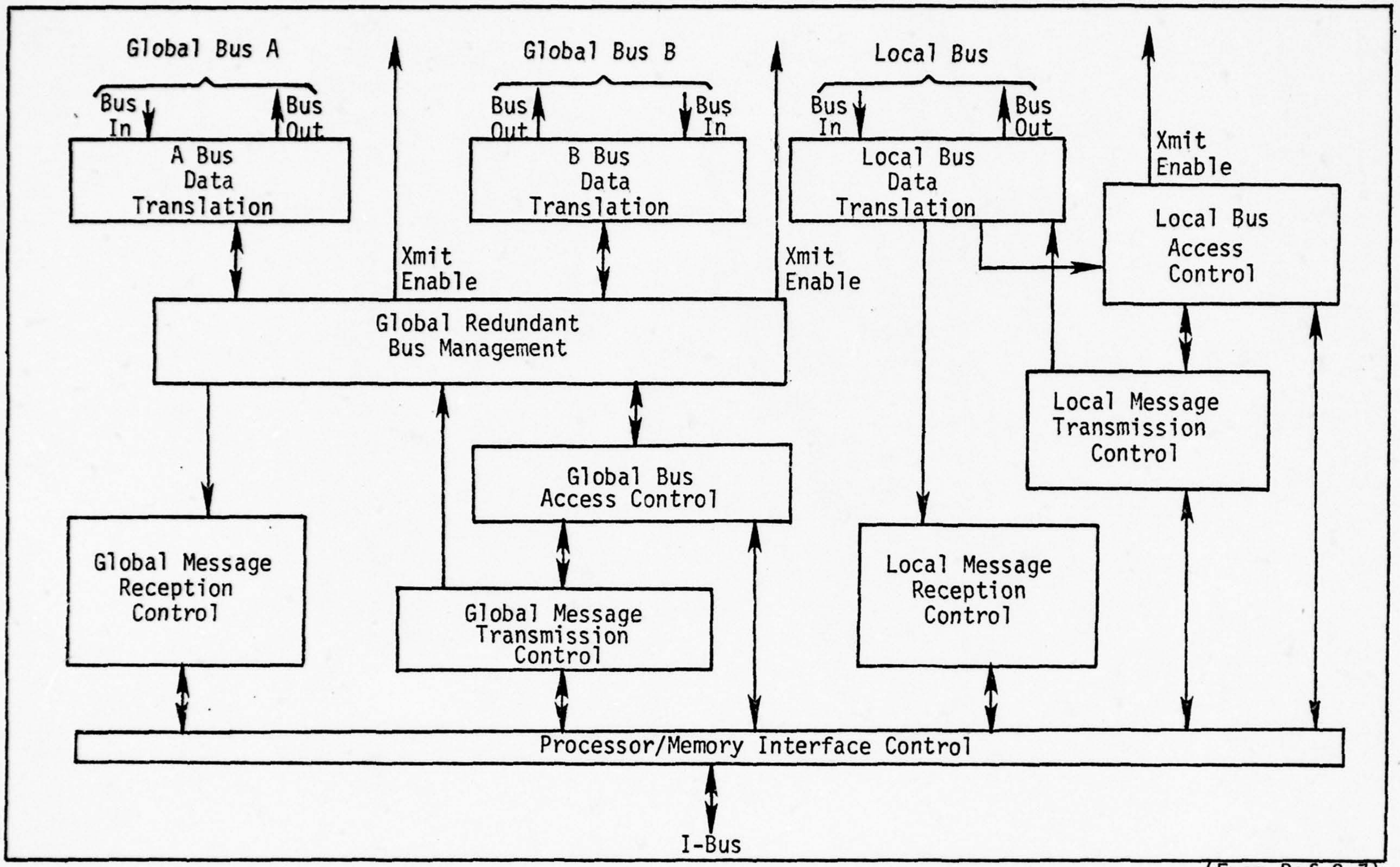
PE memory, and the Input/Output Interface Unit (IOIU). The BIU is the basic TDM data transfer interface to the PE CPU and PE memory. It translates bit serial data into parallel data words, and transfers data and status information to the PE CPU and PE memory. The PE CPU is the instruction-sequencing and data processing portion of the PE. Its computational capabilities are equivalent to present minicomputers. The PE memory provides the necessary program instruction and data storage. Memory access is local to a PE; it can be accessed by the other modules of the PE but not by another PE. Shared memory is not part of the DP/M concept. The IOIU is the interface between the PE and external devices in the avionic system. It permits digital data, command, and status information transfers between the PE and the external devices.

The four PE modules are interconnected with an Internal Bus (I-BUS). It provides intra-PE communication for data, address, and various control, status, and clock signals.

BIU Functional Description

The BIU design is based on functional modularity to provide the two levels of interfacing required (global and local). The unit is divided into six functional modules for global bus management, five of which are duplicated for local bus management. As shown in Figure 2, these functional modules are the Bus Data Translation (two required in the global area), Message Reception Control, Message Transmission Control, Redundant Bus Management (global only), Bus Access Control, and the Processor/Memory Interface Control (Ref 2).

The BIU contains the necessary functions to support a distributed round-robin bus scheme with message broadcast capability to other PE's.


Global Bus A
Bus In
Bus Out
A Bus Data Translation
Global Bus B
Bus Out
Bus In
B Bus Data Translation
Local Bus
Bus In
Bus Out
Local Bus Data Translation
Xmit Enable
Xmit Enable
Xmit Enable
Local Bus Access Control
Global Redundant Bus Management
Local Message Transmission Control
Global Bus Access Control
Global Message Reception Control
Global Message Transmission Control
Local Message Reception Control
Processor/Memory Interface Control
I-Bus

(From Ref 2:7)

Figure 2. BIU Functional Block Diagram

It contains the necessary hardware to match associatively logical input message identification headers and to vector automatically these messages into software-selectable PE memory locations. This data vectoring feature (Direct Memory Access) relieves the CPU from continually setting up and servicing the BIU hardware. This significantly speeds up the response time which is important when both the global and local buses are processing messages.

In addition to providing an interface with the global and local buses, the BIU is instrumental in aiding fault recovery in the DP/M System. Since the integrity of various system processing functions is dependent upon the correct transmittal of data between PE's, the BIU contains certain fault-detection mechanisms to monitor all communications between PE's and to alert the system upon the detection of data errors or faults.

The Problem and Approach

The TI study on the DP/M concept was completed in February 1975. The Air Force Avionics Laboratory then modified the TI functional design and issued a preliminary design specification (Ref 3). In 1976, a preliminary design of the BIU was made in a Master's thesis (Ref 2). In the TI study, the BIU was to be realized with random logic hardware. However, the feasibility of using bipolar bit-slice microprocessors for emulating the hardware functions of the BIU was demonstrated in Reference 2.

The work done in Reference 2 is primarily a feasibility study, and as such the design presented does not provide the detail required to go directly into system implementation. Devices from the Am2900 Bipolar

Microprocessor Family were used in the design, and there were several areas in the application of the chip set that could not be covered thoroughly because of lack of detailed specifications and functional descriptions on these new devices. Functional flowcharts for the BIU were developed, and the design provides a basic configuration on which a final design of the BIU can be based.

The objective of this thesis is to perform a complete design of the BIU using the Am2900 Bipolar Microprocessor Family. In addition to the hardware design, the thesis includes the functional flowcharts for all the BIU microroutines, the register transfer language (RTL) description for each of the microroutines, and the microcodes (in Mnemonic form) for these microroutines.

The development of the thesis involved seven major tasks. The first task was to study the functions of the BIU and to define the detailed BIU requirements. The second task was to evaluate and to modify or add to the functional BIU microroutine flowcharts that are presented in Reference 2. The third task was to develop a register layout for the BIU. The fourth task was to research the Am2900 Bipolar Microprocessor Family. The fifth task was to design the BIU hardware. As part of this task, the microword was developed and a decision was made to divide the global and local functions into separate units. The sixth task was to develop the RTL description for the BIU microroutines, and the seventh task was to develop the microcodes for the microroutines.

Thesis Outline

The body of this report represents a sequential description of the tasks performed in designing the BIU. Chapter II and Appendices A and B provide a detailed description of the BIU functions. Chapter III details the hardware design. Chapter IV describes the microword and discusses the microroutines and microcodes presented in Appendices B and C. Chapter V summarizes the work done, identifies additional work that needs to be accomplished, identifies departures from the BIU specification, and discusses recommendations for system improvement.

II BIU System Description

Introduction

This chapter provides a general description of the BIU functions as specified in Appendix C, Bus Interface Unit Specification, of Reference 3. Departures from the specification, which were required during the design, are identified in the discussion.

The BIU provides the interface between a PE and any of three data buses--a redundant pair of global buses and a local bus. As stated in Chapter I, the BIU performs the following primary functions:

a. Bus data translation

b. Message reception control

c. Message transmission control

d. Bus access control

e. Redundant bus management (global only)

f. Processor/memory interface

With the exception of the redundant bus management function, the above functions are identical for the global and local bus interfaces. Therefore, the descriptions that follow apply to both global and local bus interfaces. Prior to the functional description, the global/local message structure will be discussed.

Global/Local Message Structure

The message routing procedure of the DP/M System requires that each transmitted message be recognized by all PE's on the bus. To accomplish this, each message transmitted on the bus is preceded with a message header which identifies the message. Each potential receiver on the bus actively checks the message header to determine if the message is one which is to be received. A Message Identity Associative Address technique is used to decode the message header and identify the messages to be received.

Due to considerations of physical distribution of bus wiring, distance variations between PE's (bus lengths up to 300 feet), and to conform to the data code specified in Military Standard 1553, Manchester II binary data encoding is used for global/local communications. The message synchronization signals and data format are shown in Figure 3. Unique invalid Manchester waveforms are used for synchronization signals so that synchronization information can be discriminated from data patterns (Ref 1:40).

The message format of the DP/M System is shown in Figure 4. The Message Header Synchronization (MHS) denotes the start of a message (required for Manchester II decoding). The message header (MH), which has the same length as the basic data word (17 bits), contains all the message routing and control information. It is divided into four fields. The 1-bit High Priority Message field (H-field) alerts the message recipient(s) that the message requires immediate attention. The 5-bit General Control field (GC-field) allows specification of special control/status information which further defines or alters the interpretation of the message. The 10-bit Message Identify field (MID-field) specifies the identify of the associated message data. This field specifies the transmitter/receiver link during message data transmission. The Parity field (P-field) determines odd-parity for the MH. The MH fields are shown in Figure 5.

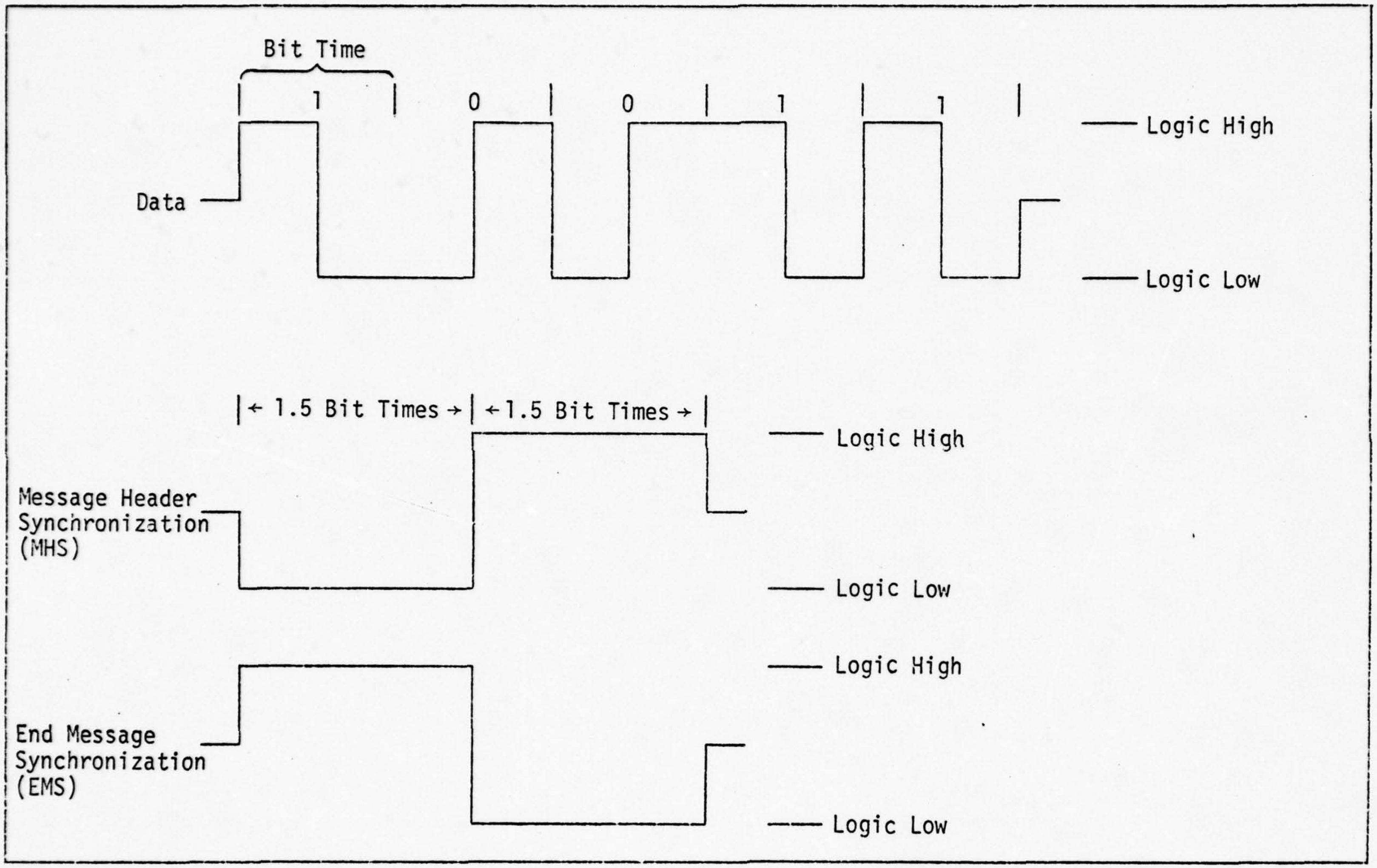


Figure 3. Manchester II Data/Synchronization Signal Waveforms (From Ref 1:41)

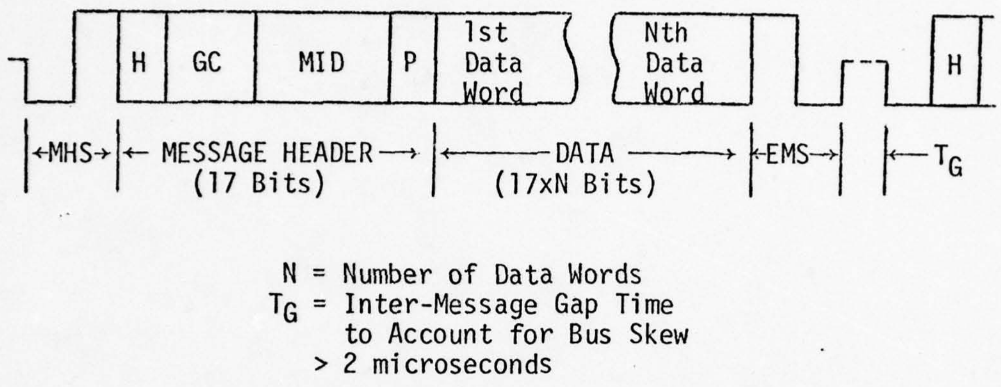


Figure 4. Message Format

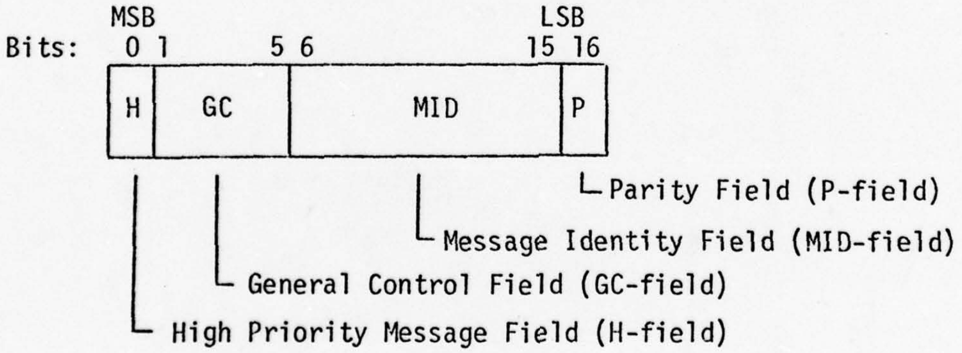

Bits:
MSB
0 1
5 6
LSB
15 16
H
GC
MID
P
Parity Field (P-field)
Message Identity Field (MID-field)
General Control Field (GC-field)
High Priority Message Field (H-field)

Figure 5. Message Header Fields

The actual data of the message immediately follows the MH. Message transfers are block oriented; accordingly, the message data content is organized into fixed-length, 16-bit data words. A parity bit is added to each data word, resulting in an overall 17-bit message data word length. Each message may contain a variable number of data words. Global bus messages, however, can be optionally limited to a maximum length of eight data words to guarantee global communication responsiveness. The unique length of each message is placed in main memory during system implementation (Ref 1:51).

Message transmission termination is denoted by the End Message Synchronization (EMS). This signal is used as the dynamic cueing mechanism which activates and controls sequencing of the distributed bus control logic throughout the DP/M System. If a PE has no data to transmit when given bus control, it "passes on" the bus control by emitting a "zero-length" message represented by an isolated EMS signal (Ref 1:51).

Bus Data Translation Function

The portion of the BIU which performs the bus data translation function was named the Bus Translation Unit (BTU) in Reference 2. The Air Force Avionics Laboratory is working on its design; accordingly, the BTU design was not included in this design effort.

The BTU is responsible for providing the interface between the global/local serial data buses and the parallel-data oriented BIU. During message reception, the BTU detects the synchronization signals MHS and EMS and notifies the BIU of an incoming message or of message termination. It converts the bi-phase encoded serial data into parallel binary data which is placed in the Input Data Register. It notifies the other parts of the BIU that a word has been received (WR signal) when its Input Data Bit Counter counts the last data word bit, bit 17. During this time, the BTU also performs error detection functions. It notifies the BIU of Manchester II encoding errors, parity errors, and word length errors.

During message transmission, the BTU converts the parallel binary data in the Output Data Buffer (Output Data Register in the BIU Specification) to bi-phase encoded serial data and appends MHS, parity bits, and EMS to the outgoing message. The BTU transmits the outgoing message when the PE's allocated time slot arrives on the bus. If the BTU is not ready to transmit or does not have anything to transmit, it passes control of the bus by immediately issuing an EMS.

Also included in this funcitonal area are two "watchdog timer" mechanisms. The Bus Quiescence Watchdog Timer (BQWT) monitors the presence of data transfer activity on the bus. If a "no-activity" time of 50 microseconds occurs, an interrupt stimulus is generated and bus access is disabled. The Bus Quiescence Watchdog Timer can be indirectly enabled or disabled by its associated interrupt mask, which is under PE CPU control.

The Bus Dominance Watchdog Timer (BDWT) monitors the length of each message on the bus. If a PE transmits a message of more than eight words, an interrupt stimulus is generated and bus access is disabled. Like the Bus Quiescence Watchdog Timer, the Bus Dominance Watchdog Timer can be enabled or disabled by its associated interrupt mask, which is under PE CPU control.

The signal transfers between the BTU and the other parts of the BIU are shown in Table I.

TABLE I

BUT Interface Signals

Signals from BTU to BIU
Message Header Synchronization
Word Received
Parallel Data
Message Encoding Error Interrupt
Message Parity Error Interrupt
Message Word Length Error Interrupt
End Message Synchronization (Received)
End Message Synchronization (Transmitted)
Bus Quiescence Interrupt
Bus Dominance Interrupt
Signals from BIU to BTU
Parallel Data
Output Pending
Transmit Message Command
Switch to Alternate Bus Command
Clear Message Encoding Error Interrupt
Clear Message Parity Error Interrupt
Clear Message Word Length Error Interrupt
Clear Bus Quiescence Interrupt
Clear Bus Dominance Interrupt
Disable Bus Quiescence Interrupt
Disable Bus Dominance Interrupt

In the BIU Specification, the serial-to-parallel data conversion, parallel-to-serial data conversion, and the watchdog timer mechanisms are part of the message reception control function, message transmission control function, and bus access control function respectively. These tasks have been included in the BTU by the Air Force Avionics Laboratory.

Message Reception Control Function

After the message header (first message word) has been assembled in parallel-data form, the message header is transferred from the BTU Input Data Register to the Input Data Register. If a message header error (word length, parity, or encoding error in the message header word) is detected during the assembly and transfer of the message header, a message header error interrupt stimulus is generated. The complete message is then ignored and further BIU message reception activity is deferred until the next message header is detected. If there are no message header errors, the BIU decodes the message header to determine if the message is for the PE. This is accomplished by extracting the six most significant bits of the Message Identity field, appending a leading 2-bit binary constant to generate a BIU PROM address which is placed into the PROM Address Register, PAR, (Figure 22). This address is used to access a word in a 64-word area defined as the Message Identify Associative Match MAP (MIAMM). Each MIAMM word consists of 16 bits and each bit is associated with a unique message header (64 words x 16 bits represent 1,024 unique messages). The appropriate response bit (RB) is determined by the binary value of the four least significant bits of the Message Identity field. The response bit of the MIAMM word is tested and if "set," the message is received. If the response bit is "not set," the message is not relevent to the PE, and the remainder of the message is ignored and further BIU message reception activity is deferred until the next message header is detected (Ref 1:64). The message identity operation is illustrated in Figure 6.

The BIU Specification requires that the MIAMM be located in the PE memory. This design places the MIAMM in the BIU PROM to increase BIU speed and to reduce traffic on the I-bus. Further discussion of this change is presented in Chapter V.

Following message recognition, the BIU places the message header into a Message Header Input Queue, a first-in-first-out modulo-8 queue located in PE memory. This action provides automatic hardware queue posting of input message identities, which are subsequently scanned and processed by the PE CPU local executive software at its convenience and throughput capability. This operation eliminates interrupt response execution time overhead which would otherwise be incurred after each individual message reception (Ref 1:64).

The address of the Message Header Input Queue is obtained from an Input Queue Pointer which is a 3-bit modulo-8 counter appended with a leading 13-bit constant. The Input Queue Pointer is incremented after each message input. The message header input operation is illustrated in Figure 7.

The data words of the incoming message are transferred to the PE memory using a procedure called "message vectoring" (Ref 1:52). Basically, each identified message is passed or "vectored" to an area in PE memory (data buffer) where it is stored for later use by an applications

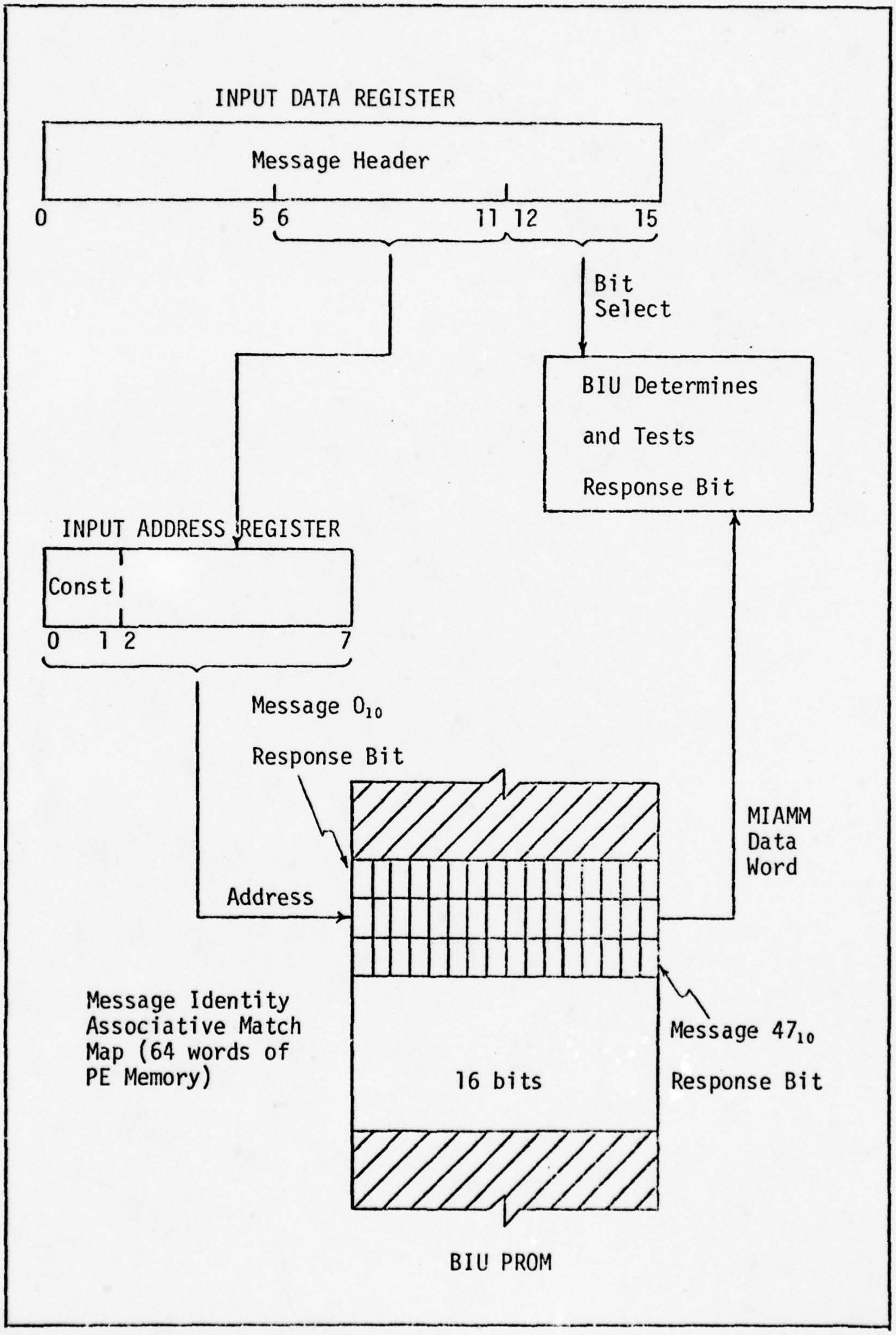

INPUT DATA REGISTER
Message Header
0
5
6
11
12
15
Bit Select
BIU Determines and Tests Response Bit
INPUT ADDRESS REGISTER
Const
0
1
2
7
Message 0_{10} Response Bit
Address
MIAMM Data Word
Message Identity Associative Match Map (64 words of PE Memory)
16 bits
Message 47_{10} Response Bit
BIU PROM

Figure 6. Message Identity Associative Addressing (Adapted from Ref 3:C-19)

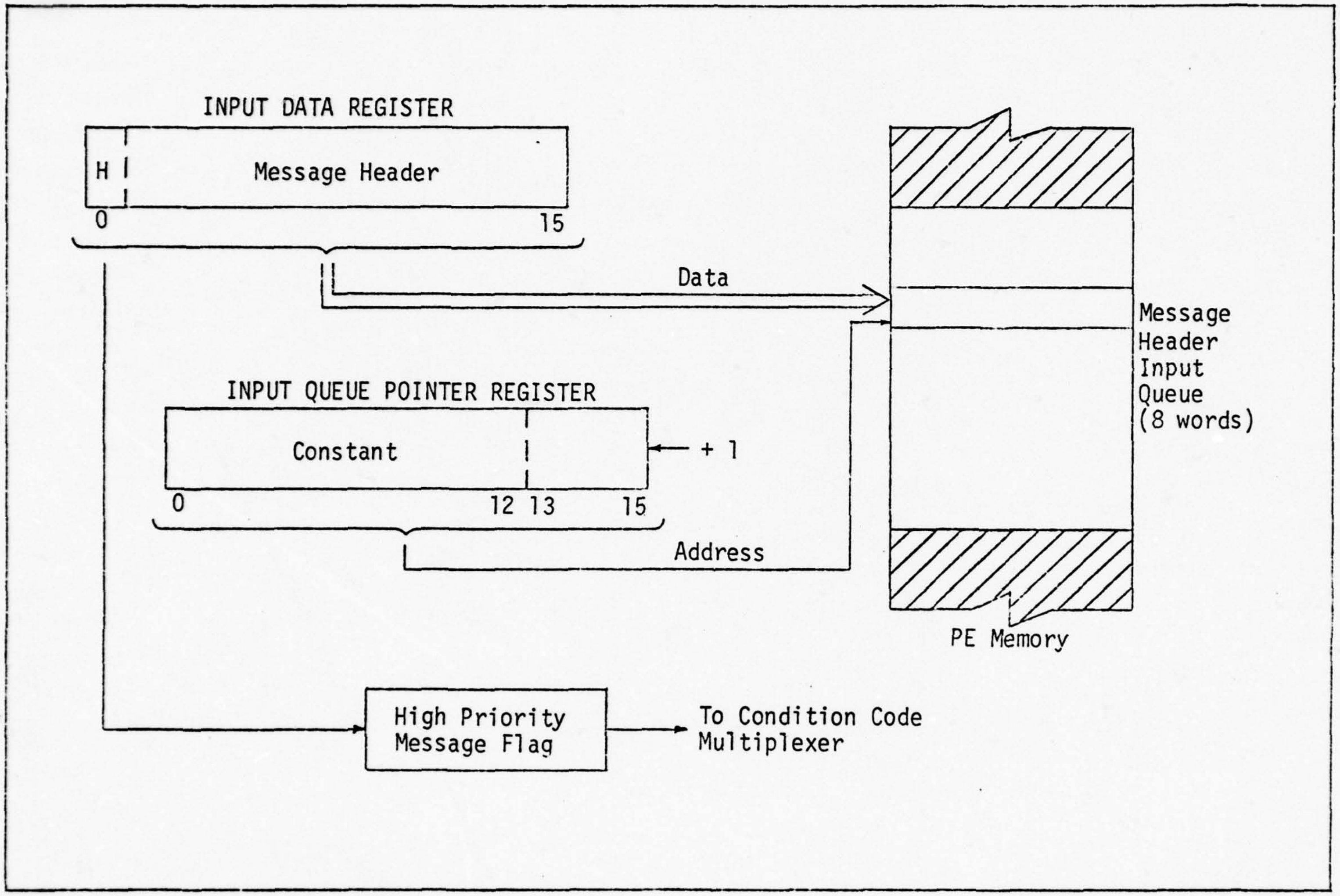


Figure 7. Message Header Input (Adapted frcm Ref 3:C-20)

software process. Data buffering is reuqired since the DP/M System operates in a loosely-coupled, asynchronous mode with respect to interprocess communication.

Message vectoring is accomplished using a hardware message vectoring technique similar to a conventional direct memory access (DMA). Conventional DMA data channels require processor (program) initialization, or "set-up" of the channel before beginning data transfers to memory. In the hardware message vectoring technique, the BIU derives a buffer storage area in PE memory from information in the message being received and then autonomously transfers the message data content to this area. No program intervention is required in setting up the input channel before each transfer (Ref 1:53).

The BIU derives the indirect address of the message buffer space by appending a leading 6-bit constant to the 10 bits of the MID-field of the message header. The derived address is placed in the Input Address Register (IAR) and a message Buffer Vector word is accessed from the PE memory. The Message Buffer Vector word is placed in the Input Address Register and is used as the PE memory address which corresponds to the first location in the message buffer space. The first word in the buffer is then accessed. This word contains the expected length of the incoming message and is placed in the Input Length Register, (ILR) (Figure 8). The Input Address Register is then incremented, preparing the BIU for the first message data word input. The BIU transfers the incoming message data words contiguously into PE memory, incrementing the Input Address Register and decrementing the Input Length Register as each word is stored. The message data input operation terminates when the Input Length Register decrements to zero.

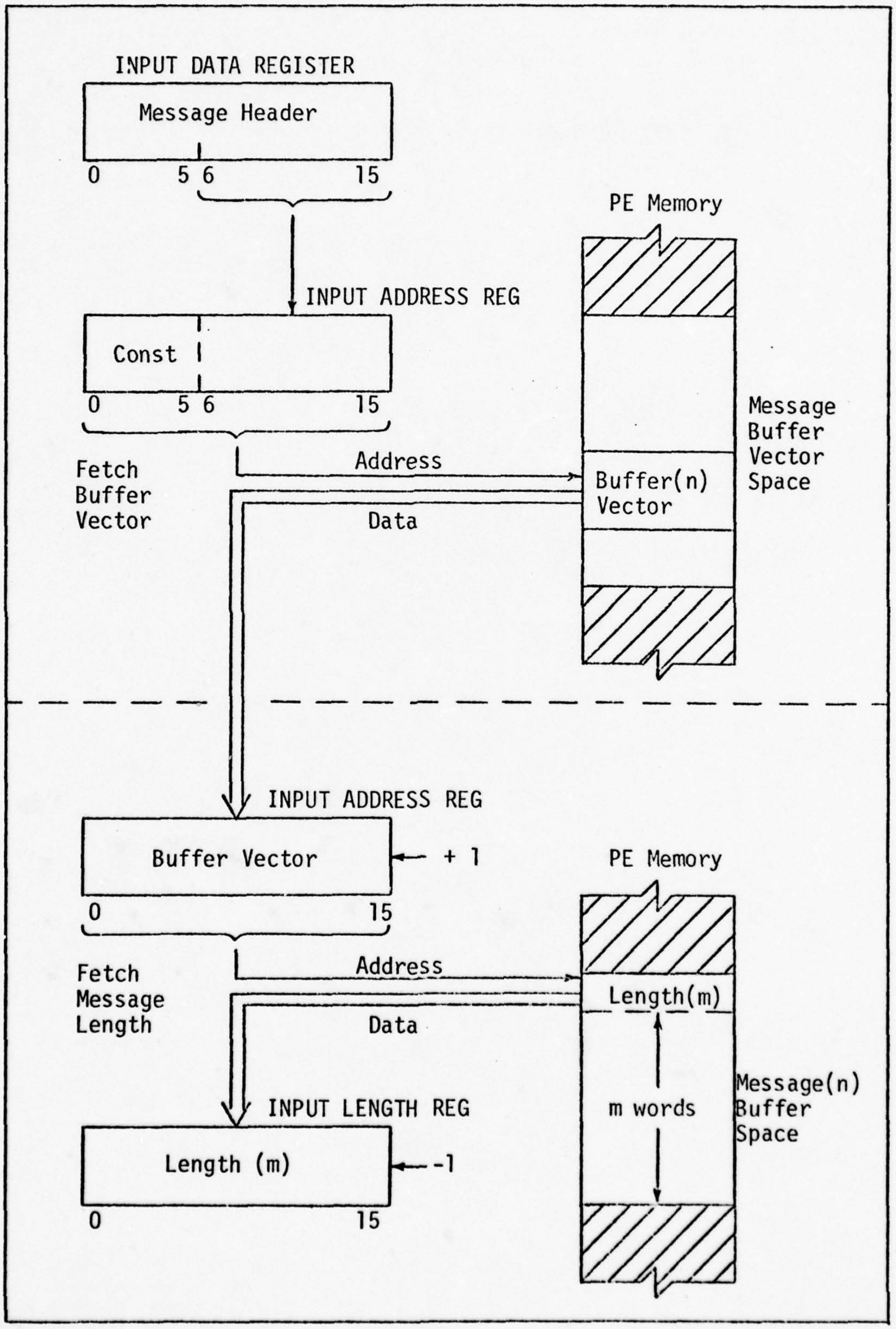


Figure 8. Message Input Initialization (Adapted from Ref 3:C-20)

Message Transmission Control Function

The message transmission control function retrieves output message data from PE memory and transfers them to the Output Data Buffer. This function effectively operates as a conventional direct memory access output channel. The operation is set up and initiated under program control by means of a Command Address Word (CAW) sent from the PE CPU which loads the Output Address Register with the beginning address of the block of data to be sent on the global/local bus. The first word accessed contains the message length which is placed into the Output Length Register. The actual output message follows, with the initial word in the message being the message header. This operation is illustrated in Figure 9. After each word is transferred from PE memory, the Output Address Register is incremented and the Output Length Register is decremented. The message transmission operation terminates when the Output Length Register decrements to zero (Ref 1:69).

The message transmission operation is under program control. Transmission is allowed when the PE CPU sets an Output Enable flag in the BIU (Ref 3:C-28).

Bus Access Control Function

The bus access control function controls PE access to the global/local bus for data transmission. Bus access control is based on a hardware implemented round-robin circular priority sequence which allows the PE to control the bus when its appropriate allocation slot is detected. Bus access control is program initialized by a Command Address Word from the PE CPU. A Command Address Word tells the BIU to load two BIU 8-bit

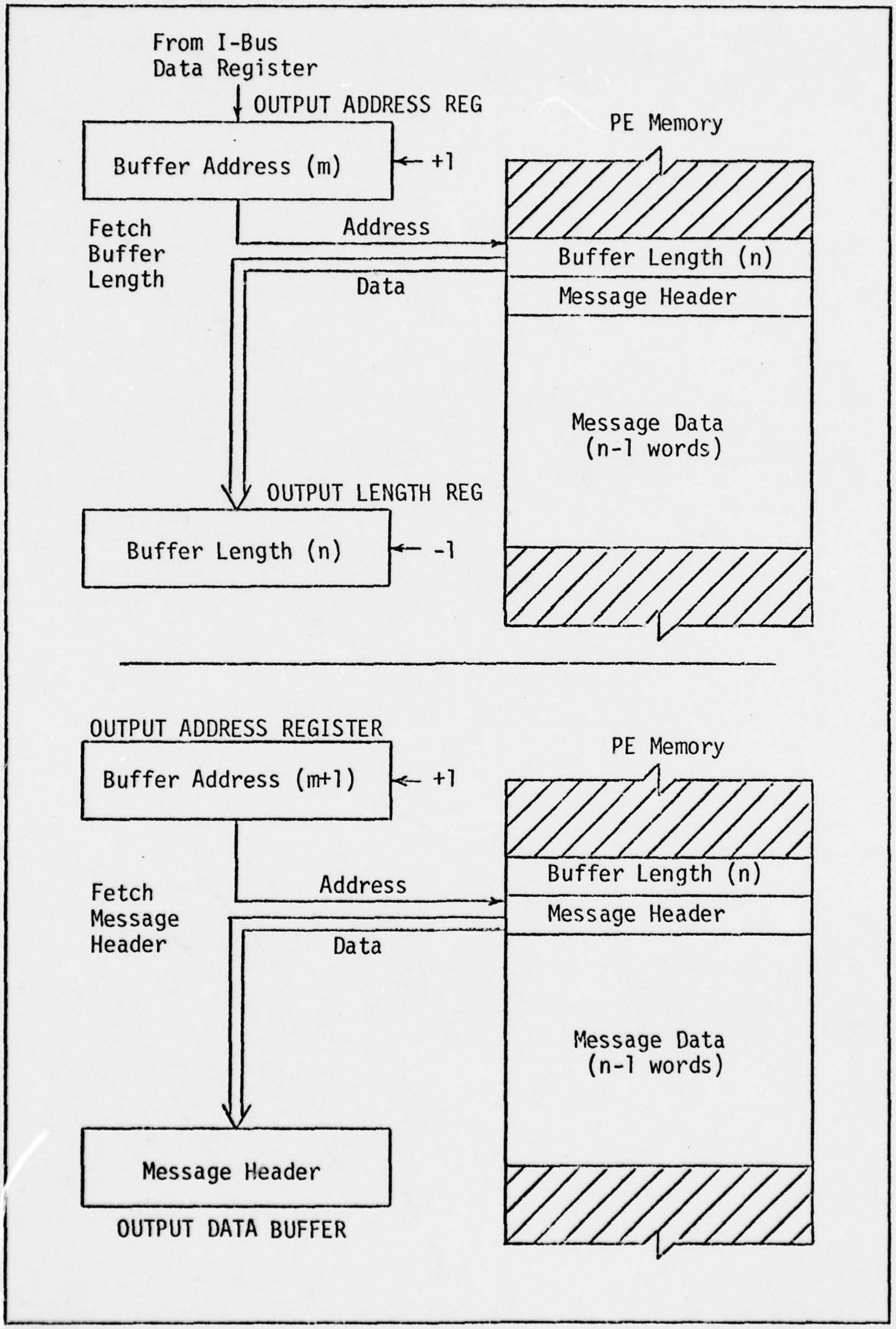


Figure 9. Message Transmission Initialization (Adapted from Ref 3:C-24)

registers, the Bus Length Register and the Bus Position Register, with the values it is sending on the I-Bus. The Bus Length Register contains the "length" in units of allocation time-slots between the PE's allocated time slot. The Bus Position Register contains the current position of the PE's allocated time slot in the bus control chain. The Bus Position Register is decremented after each EMS and when decremented to zero, it generates a Transmit Message Command signal to the BTU. The Bus Position Register is then reinitialized with the value in the Bus Length Register and the cycle is repeated.

With 8 bits, up to 256 bus allocation time slots can be defined. Since most envisioned systems will contain fewer than 32 PE's, a satisfactory degree of supercommutation (multiple allocated time slots within one round-robin cycle) is provided. This characteristic is important in reducing bus access latency time for "high-priority" bus users (Ref 1:69).

Redundant Bus Management Function

The redundant bus management function provides the necessary control elements required to interface a single BIU to dual-redundant Global buses. The redundant bus management logic continuously monitors the currently designated "alternate" bus (i.e., the bus not currently in use) for a valid switchover command issued by the Global Executive. (The Global Executive, which resides in a monitor PE, is the system monitor and scheduler.) The command is a fixed-value message, one data word in length. When the switchover command is detected, the BIU immediately terminates any message reception activity in progress by internally generating an EMS signal and a Message Encoding Error signal to the message reception function. The BTU input is transferred to the alternate bus and all future message receptions are received on this bus which now becomes the primary bus. The BIU output is under PE CPU control and is switched when the appropriate Command Address Word is issued. The switching activity can be performed as many times as directed by the Global Executive. Thus "primary" and "alternate" Global bus identities can be established dynamically and interchanged as many times as required during system operation. This switching activity is used to remove a malfunctioning PE from the primary bus (Ref 1:72; 2:29).

As stated previously, this function has been incorporated by AFAL into the BTU. Therefore, except for the function of decoding the Command Address Word for switching the output to the alternate bus, this function was not included in this design.

Processor/Memory Interface Function

The processor/memory interface performs the necessary control operations associated with providing interface protocol compatibility between the BIU and the I-Bus, which interconnects each functional unit of the PE. This function performs DMA operations and recognizes, decodes and executes Command Address Words from the PE CPU. A list of the Command Address Words is shown in Table II, page 34. This function also contains the logic required to retain and present the various BIU interrupt stimuli to the PE CPU via the I-Bus. Table III, page 35, lists the global/local interrupts generated by the BIU and sent to the PE CPU (CPU interrupts). Subfunctional responsibilities of this function are discussed in the following paragraphs. Before their discussion, a brief description of the I-Bus is given.

I-Bus Description. The I-Bus serves to transfer data between all modules of the PE (BIU, PE CPU, PE memory, and IOIU). Data transfers are handled as demand/response sequences. As such, all signals are transmitted and received between an I-Bus master device, which controls data transfer, and an I-Bus slave device, which generates or accepts data in response to the master device.

The modules of the PE compete for I-Bus access on a priority basis. the BIU is assigned the highest priority and the CPU the lowest priority.

The I-Bus consists of control lines, information lines, and general facilities lines. There are seven lines for bus control activities, three control lines and a set of four identification lines. The three bus control signals are Bus Request (BRQ), Bus Release (BREL), and Bus Grant (BGR). The Bus Request signal is used to initiate a bus master assignment, while the Bus Release signal is used to terminate the assignment. The Bus Grant line is threaded through each module (daisy chain) and is used to resolve conflicts when two or more masters request the bus simultaneously. The four identification lines, Bus Master Identification (BMID), are associated with the maintenance function, and are used to identify the module in control of the bus.

Thirty-seven lines are utilized for data transfer operations on the bus. Sixteen lines are used for address (ADDR), 16 lines for data (DATA), and the remaining lines for data transfer control. These control lines are I/O Select (IOSL), Data Receive (DRCV), Transfer Request (TRQ), Transfer Acknowledge (TACK), and Transfer Time Out (TTO).

There are three I-Bus control lines associated with CPU interrupt activities. They are Interrupt Request (IRQ), Interrupt Inhibit (INHB), and Interrupt Acknowledge (IAK). The Interrupt Acknowledge line, like the Bus Gran line, is threaded through each module in a daisy chain fashion and is used to resolve conflicts when two or more modules attempt to interrupt the CPU simultaneously.

The general facilities lines are composed of a free running clock (FCLK), a master clock (MCLK), a Master Stop (MSTP), a Clear/Reset (MRESET), power, and ground.

With the exception of the BMID, IOSL, and DRCV, all signals on the I-Bus are transmitted as complements. Positive logic is employed, i.e., a high voltage is a logic "1" (True) and a low voltage (ground) is a logic "0" (False).

The I-Bus is asynchronous. The speed of data transfers over the bus is determined by the speed of the modules interfaced to it.

Direct Memory Access Function. The BIU requests PE memory access on a cycle-stealing-by-priority basis in response to the memory requests of the message reception control and message transmission control activities. As stated previously, the BIU possess the highest memory access priority (I-Bus access) among all the modules of the PE. Within the BIU, the BIU specification establishes the following priority in decending order:

1. Global Bus input activity
2. Global Bus output activity
3. Local Bus input activity
4. Local Bus output activity

In this design, the global and local functions have been separated into separate modules. The Global BIU has the highest priority and the Local BIU has the next highest priority.

In performing direct memory access (DMA), the BIU acts as a master device on the I-Bus. It initiates bus control by issuing $\overline{\text{BRQ}}$ (BRQ low designates bus request). When it receives $\overline{\text{BGRI}}$ (Bus Grant signal into the BIU), it becomes bus master, and it blocks the Bus Grant signal from propagating to any lower priority module which may also be requesting the bus. This is done by making the Bus Grant line high beyond the BIU (Bus Grant signal out of the BIU is redesignated BGRO). After obtaining control of the bus, the BIU issues BMID and waits until the previous data transfer is completed (TRQ and TACK lines high) before it starts its data transfer operation.

For a write-into-memory operation, the BIU initiates the data transfer operation by issuing BMID, $\overline{\text{TRQ}}$ (TRQ low designates transfer request to the slave, which is memory), $\overline{\text{IOSL}}$ (IOSL low designates memory), $\overline{\text{DRCV}}$ (DRCV low designates master to write, send operation), ADDR, DATA, and $\overline{\text{BREL}}$ (BREL low designates bus release). When $\overline{\text{BREL}}$ is issued, all bus masters, including the BIU, remove their bus requests. Wtih the BIU's bus request removed, BGRO goes low (Bus Grant signal now available to all masters), and $\overline{\text{BREL}}$ is then released, allowing all masters including the BIU, to request the bus. This sequence of releasing the bus at the start of an I-Bus data transfer operation allows bus mastership resolution to be overlapped with I-Bus data transfers. The BIU performs block transfers by immediate rerequest of the bus. When a lower priority module is performing a block transfer, the BIU can always pre-empt between transfers by requesting the bus when BREL goes high.

When the BIU issues $\overline{TRQ}$, $\overline{IOSL}$, $\overline{DRCV}$, ADDR, and DATA, the PE memory receives $\overline{TRQ}$ and decodes IOSL and ADDR to determine if it is being addressed. The PE memory internally delays the $\overline{TRQ}$ signal to allow for valid address data reception and worst case address decode time. The internally delayed $\overline{TRQ}$ signal and a valid address code is then used to generate a memory start signal. When the memory write cycle is complete the PE memory issues $\overline{TACK}$ (TACK low designates transfer acknowledge). The BIU receives the asserted transfer acknowledge and it removes $\overline{TRQ}$, $\overline{IOSL}$, $\overline{DRCV}$, ADDR, and DATA. When the PE memory detects the removal of $\overline{TRQ}$, it removes $\overline{TACK}$, at which time the I-Bus is relinquished to the next I-Bus master. The DMA write operation is illustrated in Figure 10.

For a read-from-memory operation, the BIU, after obtaining bus access and after previous I-Bus operations are complete, issues $\overline{TRQ}$, $\overline{IOSL}$, DRCV (DRCV high designates master to receive, read operation), ADDR, and $\overline{BREL}$. The PE memory, after it determines that it is to be read, starts a read cycle. When the data (DATA) is valid, the PE memory issues $\overline{TACK}$. The BIU receives the asserted transfer acknowledge and after a delay to assure receipt of valid data, latches DATA and removes $\overline{TRQ}$, $\overline{IOSL}$, DRCV, and ADDR. When the PE memory detects the removal of $\overline{TRQ}$, it removes $\overline{TACK}$ and DATA, at which time the I-Bus is relinquished to the next I-Bus master. The DMA read operation is illustrated in Figure 11.

In addition to the normal transfer request/acknowledge sequence, there is one way in which the DMA operation can be terminated abnormally. A Transfer Time Out (TTO) is generated by a watchdog timer on the memory controller when the BIU (or any master) attempts to address a

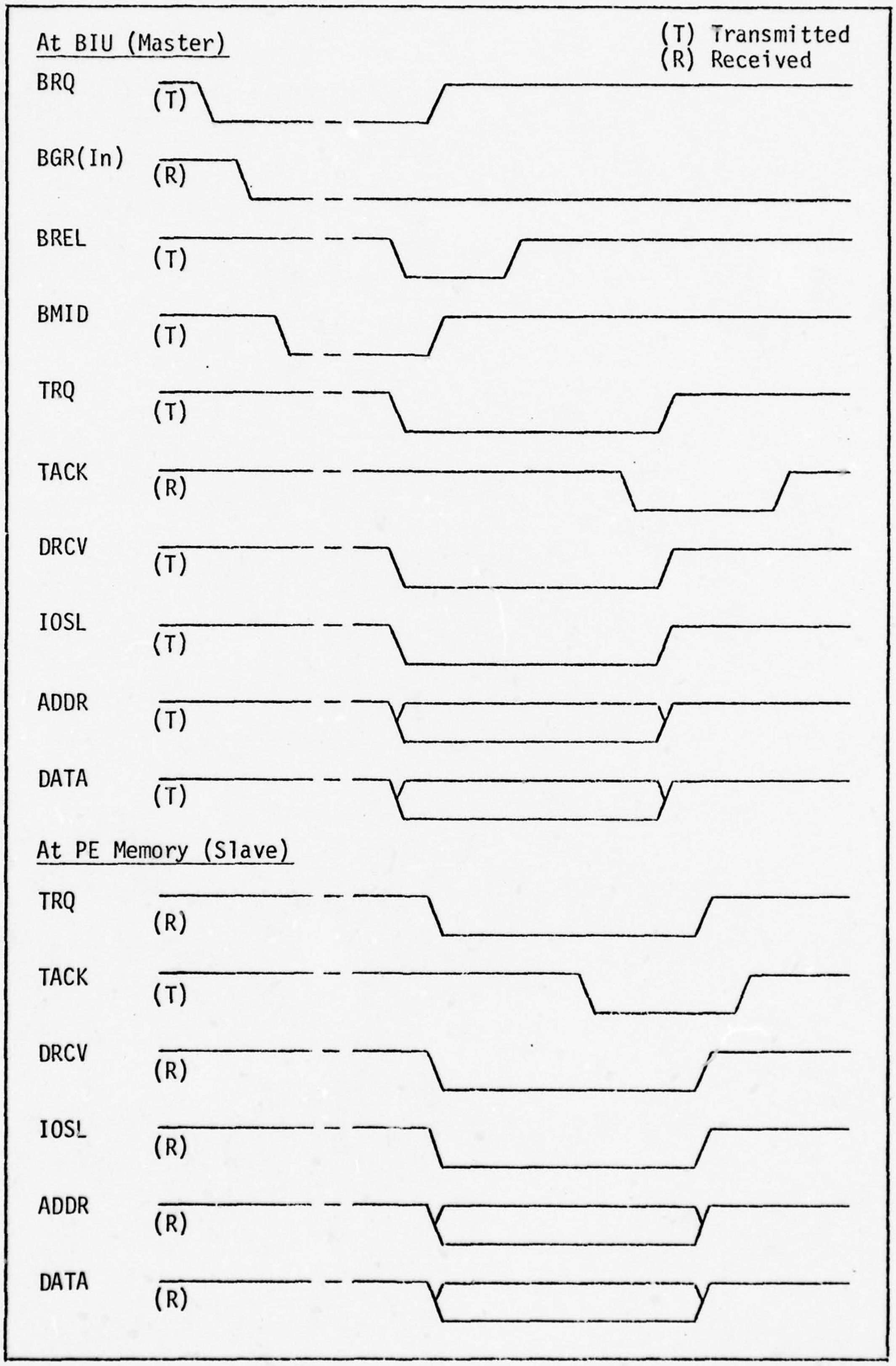


Figure 10. Timing for DMA Write Operation (Adapted from Ref 1:160)

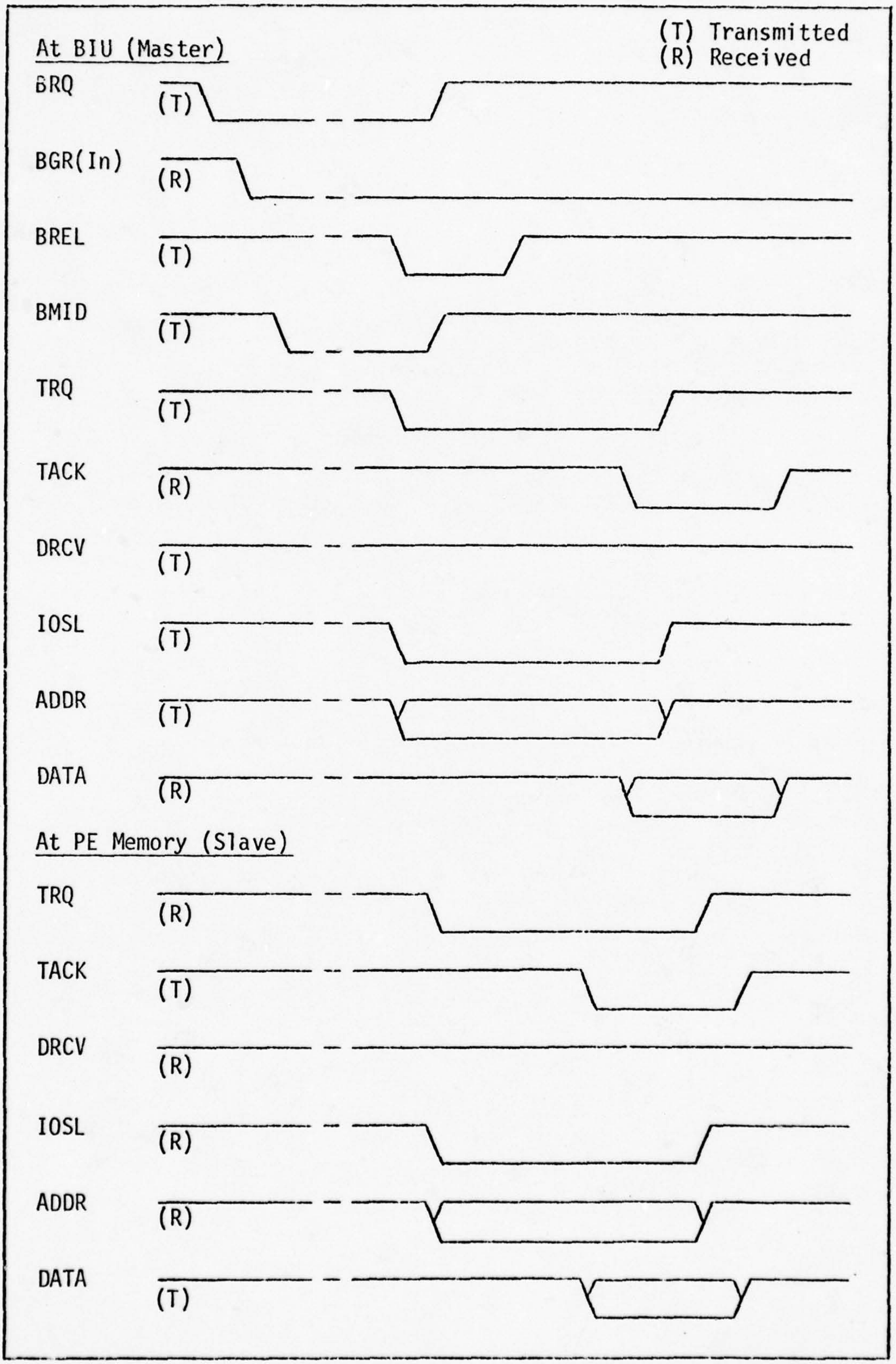


Figure 11. Timing for DMA Read Operation (Adapted from Ref 1:161)

nonexistent memory location (or slave). When the time out occurs, the BIU uses the time out signal in the same manner as a transfer acknowledge signal, except it sets the Transfer Time Out interrupt and proceeds (Ref 3:C-34, B-3 to B-6). The Transfer Time out Interrupt has been added in this design and is discussed further in Chapter V.

Program-Command Response Function. This function deals with processing program commands which the PE CPU issues in the form of Command Address Words (CAW), Table II. In performing this function, the BIU acts as a slave device on the I-Bus.

In a write-into-BIU command, the BIU receives $\overline{\text{TRQ}}$ (TRQ low designates transfer request to the slave), IOSL (IOSL high designates input/output device), $\overline{\text{DRCV}}$, (DRCV low designates master to write), ADDR (contains the CAW), and DATA. The BIU decodes these signals and if it is determined that the CAW is for the BIU, the content of the I-Bus Data Register is clocked into the BIU and Transfer Acknowledge ($\overline{\text{TACK}}$) is issued. When the PE CPU removes $\overline{\text{TRQ}}$, the BIU removes $\overline{\text{TACK}}$.

In a read-from-BIU command, the procedure is similar to the above except that DRCV (DRCV high designates master to read) is received instead of $\overline{\text{DRCV}}$. If it is determined that the CAW is for the BIU, the requested information is clocked onto the I-Bus and $\overline{\text{TACK}}$ is issued. When the PE CPU removes $\overline{\text{TRQ}}$, the BIU removes $\overline{\text{TACK}}$.

Interrupt Interface/Control Function. This function deals with CPU interrupts, interrupts generated within the BIU which are passed to the PE CPU (See Table III). This function performs program-controlled masking of each of the interrupts for the PE CPU and provides protocol compliance with the CPU interrupt-related operations.

TABLE II

BIU CAW Assignments

CAW GLOBAL/LOCAL (Hexadecimal)		WRITE/READ (DRCV/DRCV)	FUNCTION	ACTION
03	02	$\overline{\text{DRCV}}$	Output* G/L Interface Status	[IBDR] → ALU(CISR)
03	02	DRCV	Input G/L Interface Status	[ALU(CISR)] → IBDR
05	04	$\overline{\text{DRCV}}$	Activate G/L Output	1 → ALU(CISR/OAF bit) [IBDR] → OAR
07	06	$\overline{\text{DRCV}}$	Output G/L Bus Control	[IBDR/bits 0-7] → BLR [IBDR/bits 8-15] → BPR
07	06	DRCV	Input Present G/L Position	[BPR] → IBDR/bits 8-15
09	08	DRCV	Input G/L Queue Pointer	[IQPR] → IBDR
0B	0A	$\overline{\text{DRCV}}$	Enable G/L Output	1 → ALU(CISR/OEF bit)
0D	0C	$\overline{\text{DRCV}}$	Disable G/L Output	0 → ALU(CISR/OEF bit)
09	-	$\overline{\text{DRCV}}$	Switch to Alternate Bus	

*Output and Write refer to PE CPU writing into BIU. Similarly, Input and Read refer to PE CPU reading from BIU.

(Ref 3:C-37)

TABLE III

CPU Interrupts

CLASS	INTERRUPTS IN DESCENDING PRIORITY
Arm/Disarm	Global High-Priority Message Received (GHMR)
Enable/Disable	Global Bus Quiescent (GBQ)
Enable/Disable	Global Bus Dominance (GBD)
Arm/Disarm	Global Message Length Violation (GMLV)
Arm/Disarm	Global Message Header Error (GMHE)
Arm/Disarm	Global Message Word Count Error (GMWCE)
Arm/Disarm	Global Message Word Length Error (GMWLE)
Arm/Disarm	Global Message Parity Error (GMPE)
Arm/Disarm	Global Message Encoding Error (GMEE)
Arm/Disarm	Global Transfer Time OUt (GTTO)
Enable/Disable	Global Message Received (GMR)
Arm/Disarm	Global Transmission Completed (GTC)
Arm/Disarm	Local High-Priority Message Received (LHMR)
Enable/Disable	Local Bus Quiescent (LBQ)
Enable/Disable	Local Bus Dominance (LBD)
Arm/Disarm	Local Message Length Violation (LMLV)
Arm/Disarm	Local Message Header Error (LMHE)
Arm/Disarm	Local Message Word Count Error (LMWCE)
Arm/Disarm	Local Message Word Length Error (LMWLE)
Arm/Disarm	Local Message Parity Error (LMPE)
Arm/Disarm	Local Message Encoding Error (LMEE)
Arm/Disarm	Local Transfer Time Out (LTTO)
Enable/Disable	Local Message Received (LMR)
Arm/Disarm	Local Transmission Completed (LTC)

In the present DP/M scheme, the BIU does not simultaneously relay all pending CPU interrupts to the PE CPU. As shown in Table III, the interrupts are prioritized. They are serially relayed to the PE CPU with the highest priority interrupt being relayed first. Before relaying an interrupt, the BIU must first determine if the PE CPU is presently accepting the specific interrupt. This is done by checking an interrupt mask which the PE CPU sets under program control. The PE CPU places the mask in the Interrupt Status Register, ISR (Figure 12). In addition to the interrupt mask, the BIU Specification requires three other bits of information in the register: an Output Pending Flag and Output Active Flag, both of which the BIU sets, and an Output Enable Flag which the PE CPU sets.

If the mask for a particular pending interrupt is set (Interrupt not allowed), the BIU does one of two things depending on the class of the interrupt. If the interrupt is of the enable/disable class, the interrupt is discarded; the interrupt stimulus is cleared and the BIU proceeds with its other operations. If the interrupt is of the arm/disarm class, the interrupt is retained by the BIU until the interrupt's associated mask is cleared by the PE CPU, at which time the interrupt is processed to the PE CPU (Ref 3:C-35 to C-38).

In relaying an interrupt to the PE CPU, the BIU first checks the status of the I-Bus interrupt Inhibit (INHB) line, which is used by the PE maintenance panel to inhibit all PE modules from interrupting the PE CPU when the PE CPU is operating in the maintenance panel mode. If there is no Inhibit signal (INHB high), the BIU proceeds with interrupting the PE CPU by issuing an $\overline{\text{IRQ}}$ on the I-Bus (IRQ low designates interrupt

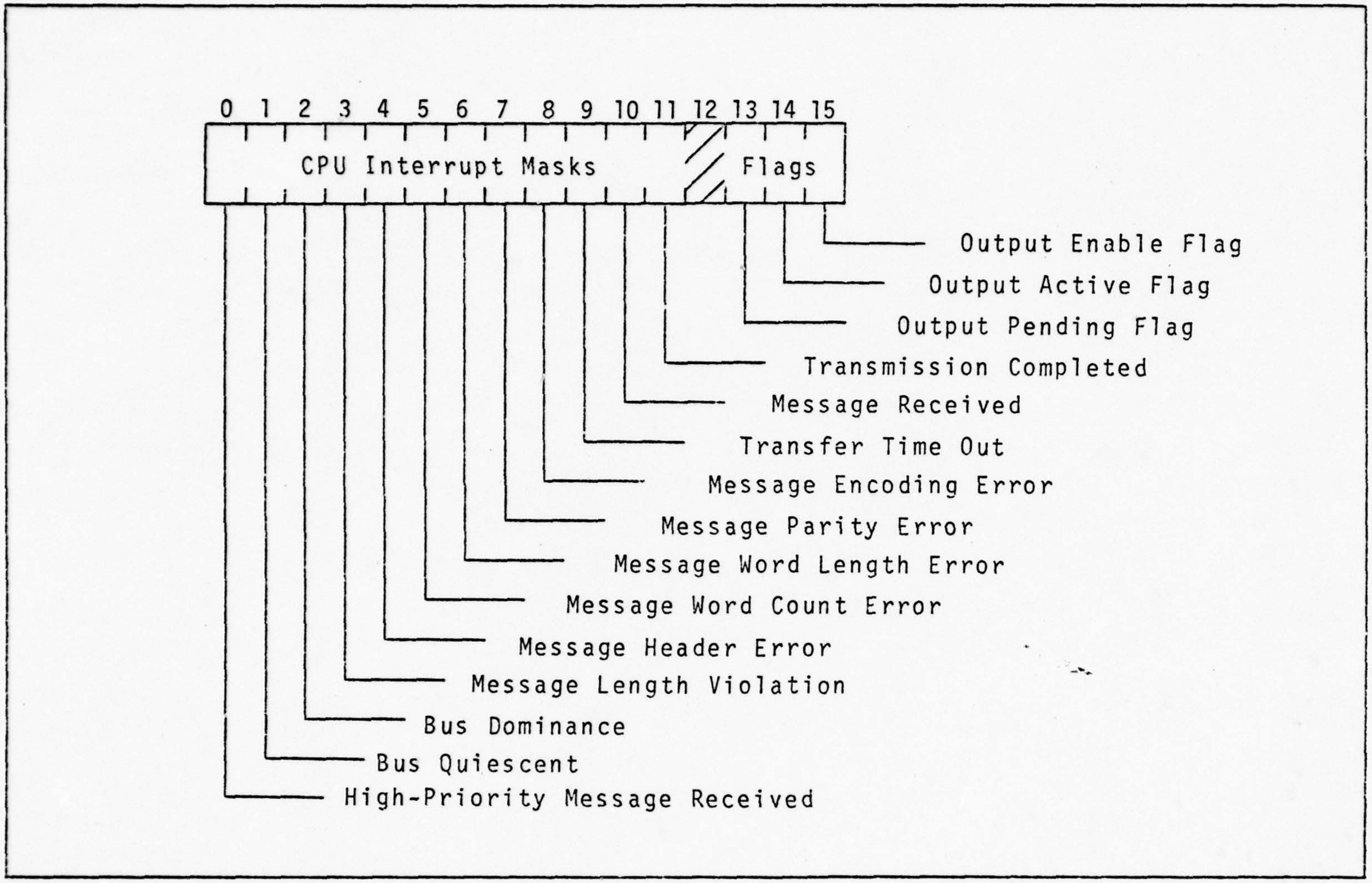


Figure 12. Global/Local Interrupt Status Register

request). When the PE CPU completes the execution of its current macro instruction, it detects an interrupt, then gains control of the I-Bus and honors the interrupt by issuing $\overline{\text{IAK}}$ (IAK low designates interrupt acknowledge). When the BIU receives $\overline{\text{IAK}}$, it inhibits the $\overline{\text{IAK}}$ signal from propagating to the lower priority modules. It then removes $\overline{\text{IRQ}}$ and places the interrupt trap vector address on the data lines, DATA. After at least a 150 nano second delay, the BIU indicates that it has placed the trap vector address on the lines by issuing $\overline{\text{TACK}}$ (TACK low designates transfer acknowledge). The processor latches the trap vector address into an internal register and then removes $\overline{\text{IAK}}$. When the BIU detects the removal of $\overline{\text{IAK}}$, it completes the interrupt operation by removing the trap vector address from the data lines and removing $\overline{\text{TACK}}$ (Ref 3:B-6). The interrupt operation is illustrated in Figure 13.

The BIU Specification requires the BIU to provide the PE CPU with an interrupt level trap vector address instead of the trap vector address for the specific interrupt. The specification change to provide the specific interrupt trap vector address is discussed in more detail in Chapter V.

Summary

This chapter presented the functional description of the BIU. Functional requirements definition was the first step in the design of the BIU. The next step was to define the hardware and microword required to implement the BIU functions. These are presented in the next chapter.

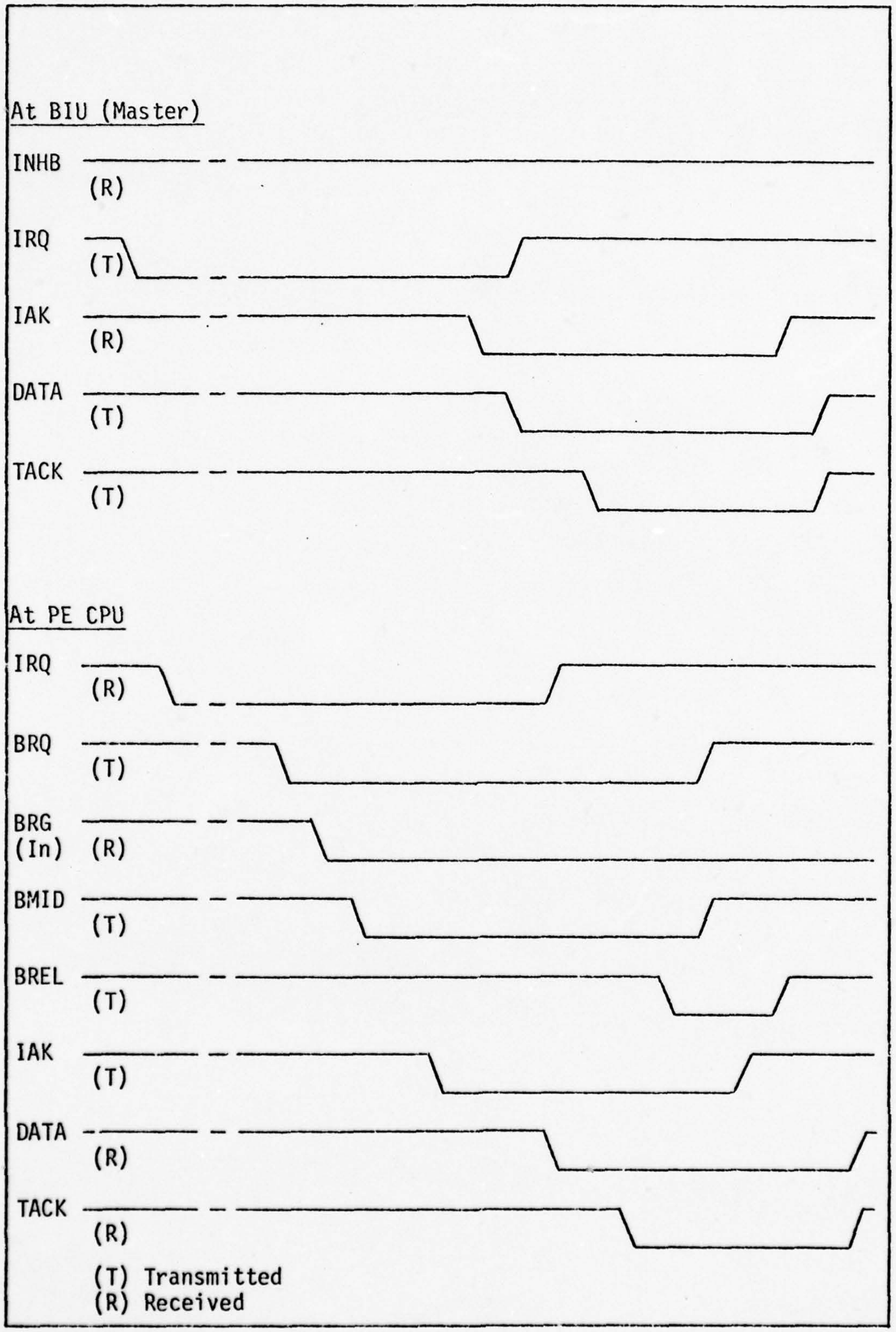


Figure 13. Timing for Interrupt Operation (Adapted from Ref 1:165)

III BIU DESIGN

Introduction

The BIU designed is a laboratory test model to be used in demonstrating the DP/M concept. This design uses a bipolar bit-slice microprocessor, specifically the Am2900 Bipolar Microprocessor Family, for emulating the BIU functions. In the TI DP/M report, the BIU was to be realized with random logic hardware, and the unit was divided into functional modules (Ref 1:72-94). The feasibility of utilizing a bipolar bit-slice microprocessor for the BIU has since been demonstrated (Ref 2). With a microprocessor, it is not possible to physically maintain the functional modules proposed in the TI DP/M report, but the functional modularity is maintained in the software. Table B-1 lists the microroutines that were developed to perform the primary functions of the BIU.

Although this design uses a bit-slice microprocessor as proposed in Reference 2, a different design philosophy is used. In this design, more functions are accomplished in hardware to meet the timing requirements of the system. In the design presented in Reference 2, most of the BIU operations are accomplished in software. The philosophy was that software implementation provided greater flexibility during initial system construction and testing. Most changes could be made in software which would not require major engineering redesign. If the required speed could not be obtained, the laboratory DP/M system could be slowed down to demonstrate the feasibility of the DP/M system. Since the system was envisioned for the 1980 time-frame, it was felt that the state-of-the-art would advance sufficiently at that time to provide a microprocessor to meet all BIU timing requirements (Ref 2:11, 12, 76).

A more convincing demonstration of the DP/M concept can be made if the laboratory system operates at its specified speed. In this design, the use of a microprocessor is maintained because of the flexibility feature that it provides. To meet the speed requirements, it was necessary to implement some of the functions in hardware. In addition, it was necessary to have two separate BIU's, one for global operations and one for local operations (Ref 1:93; 2:73). The BIU is required to simultaneously transmit and receive on one global bus, monitor the alternate global bus, and transmit and receive on the local bus. In addition, communication with the PE CPU and PE Memory must be performed on a common (shared) I-Bus. Since it appeared to be impossible to perform these functions simultaneously, separate global and local BIU's are proposed.

With the exception of the redundant bus management function, the Local BIU is identical to the Global BIU. In the remainder of the chapter, only the Global BIU will be described. The hardware design and software apply to both units. For the Local BIU, where the term "Global" is used, the term "Local" can be substituted.

Because the global bus interface operations and internal bus interface operations are asynchronous operations (essentially demand/response operations) the BIU is designed as an interrupt driven microprocessor. Its detailed description is presented in this chapter. First a general description of the Am2900 Family is given, then the BIU architecture is discussed. The detailed system description begins with the Control

Section then continues with the Data Manipulation Section, I-Bus Interface Section, and the Global Bus Interface Section. Descriptions of the microroutines required to operate the BIU are presented in Appendix B.

Am2900 Bipolar Microprocessor Family General Description

The Advanced Micro Devices Incorporated Am2900 Bipolar Microprocessor Family is a group of large scale integrated (LSI) circuits designed for use in microprogrammed computers and controllers. They consist of 4-bit bipolar bit-slice devices, each device designed to be expandable and flexible to allow construction of a wide variety of machine architectures. Designed to be fundamental system building blocks, the Am2900 Family consists of circuits designed to perform data manipulation, microprogram control, macroprogram control, priority interrupt, direct memory access, input/output control, memory control, and front panel control. Utilizing Low-Power Schottky TTL Technology, these devices are very fast. See References 4, 5, 6, and 7 for technical data on the Am2900 devices used in this design.

BIU Architecture

The block diagram of the BIU is shown in Figure 14. The BIU is a microprogrammed processor. Each BIU component performs a basic function and is driven by a set of control lines from a microinstruction in the Pipeline Register. The microinstruction is obtained from the Microprogram PROM.

The BIU can be divided into four major sections. First is the Global Bus Interface Section, which consists of the Bus Translation Unit (BTU), Bus Length Register (BLR), Bus Position Register (BPR), and

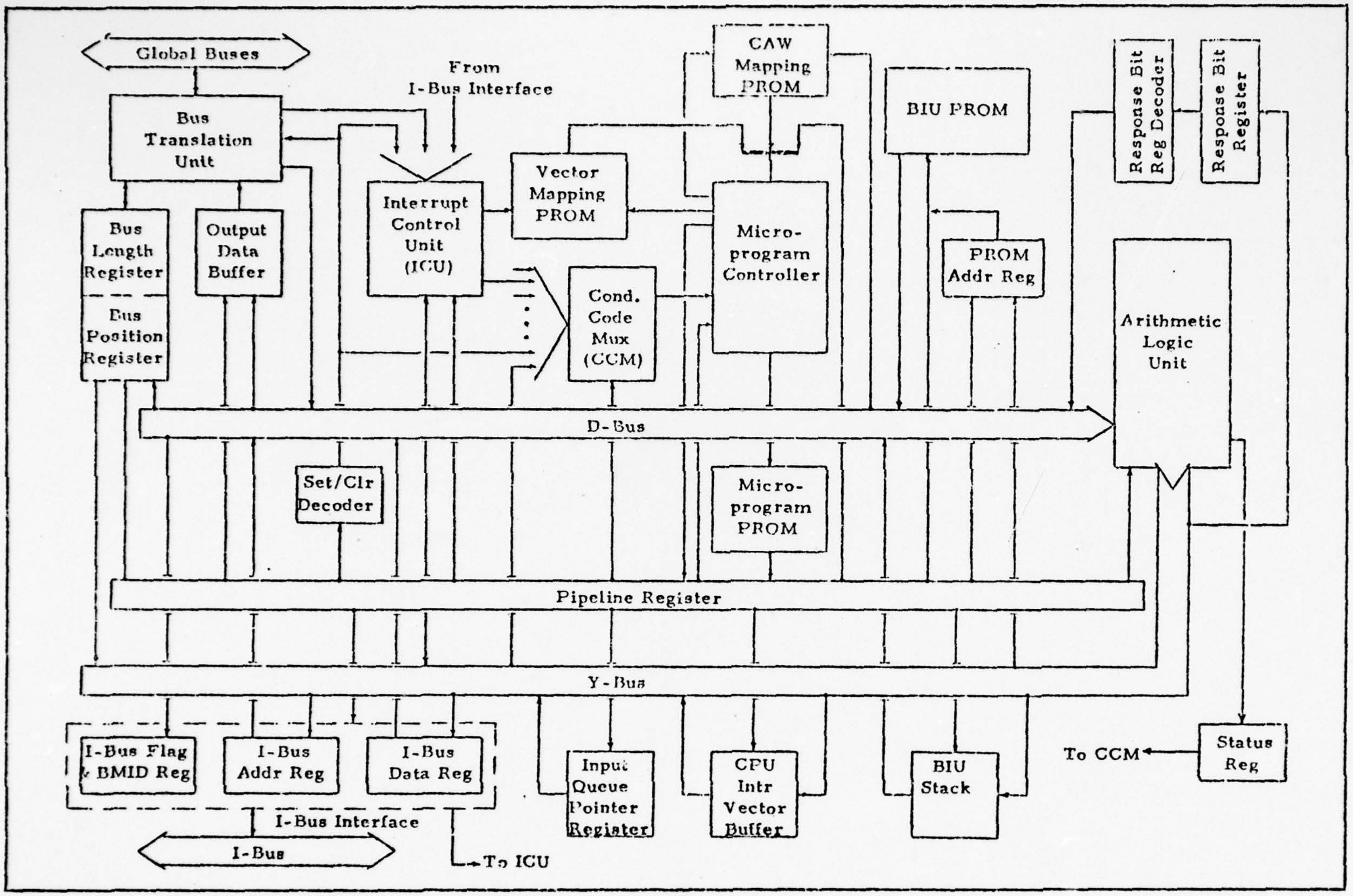


Fig. 14. BIU Block Diagram

The Output Data Buffer (ODB). As the name implies, this section interfaces the BIU with the two global buses.

The second section is the Control Section (Figure 15), which consists of the Interrupt Control Unit (ICU), Vector Mapping PROM (VMP), Condition Code Multiplexer (CCM), Condition Code Flags (CCF), Condition Code Latch (CCL), CAW Mapping PROM (CMP), CAW Mapping PROM Register (CMPR), BIU Stack (STK), Microprogram Controller (MC), Microprogram PROM (MP), Pipeline Register (PR), and the Set/Clear Decoder (SCD). This section determines which microroutine is to be executed. Specifically, it selects the microinstruction to be executed which contains the bits to control each of the data handling components in the BIU.

The third section is the Data Manipulation Section, which consists of the Arithmetic Logic Unit (ALU), Status Register (SR), BIU PROM (PROM), PROM Address Register (PAR), and the Response Bit Register (RBR) and decoder (RBRD). Data is processed by moving it from a register onto the D-Bus and into a working register in the ALU, performing the required operations on the data, and outputting the results onto the Y-Bus.

The four section is the I-Bus Interface Section, which consists of theBMID register (BMIDR), I-Bus Flag Register (IBFR), I-Bus Address Register (IBAR), I-Bus Data Register (IBDR), I-Bus Slave Control Logic (ISCL), I-Bus Master Control Logic (IMCL), and the CPU Interrupt Control Logic (CICL). Two other components associated with this section are the Input Queue Pointer Register (IQPR), and the CPU Interrupt Vector Buffer

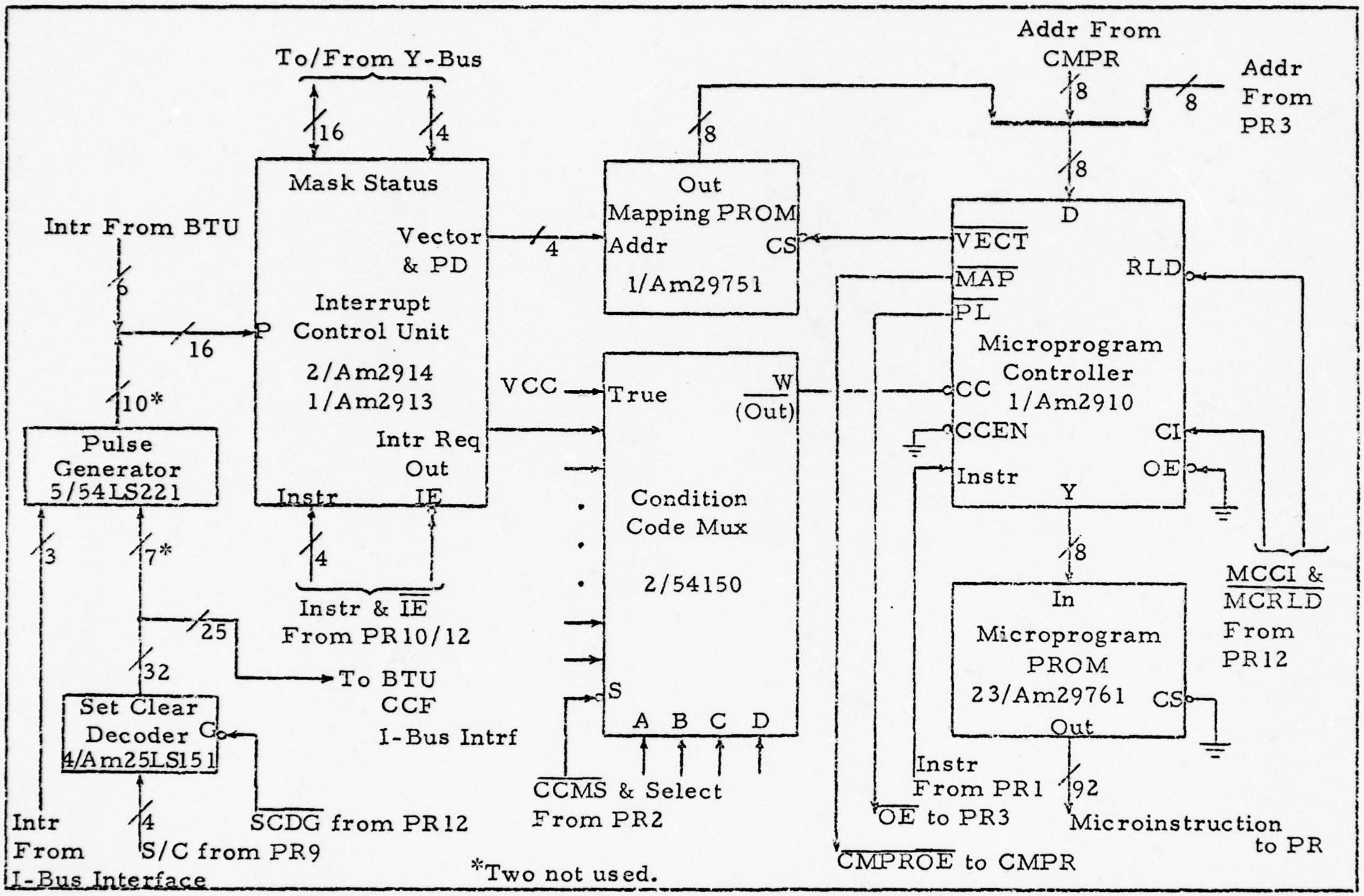


Fig. 15. BIU Control Section (Excluding CAW Decode Circuits, BIU Stack, and Pipeline Register)

(CIVB). As the name of the section implies, this section interfaces the BIU with the I-Bus.

There are two major buses internal to the BIU; a 16-bit wide D-Bus, primarily used for inputting data into the ALU, and a 16-bit wide Y-Bus, primarily used for outputting data from the ALU. All data transfers between registers are done via these two three-state buses.

First level pipelining is employed to improve the BIU performance. This is a technique in which the microinstruction for the next cycle is fetched from the Microprogram PROM on an overlapping basis with the execution of the current microinstruction. In this technique, the microprogram fetch, the current microinstruction being executed, and the results of the previous microinstruction are simultaneously available with respect to each other. The key hardware components are the Pipeline Register at the output of the Microprogram PROM which contains the microinstruction currently being executed and the Status Register at the output of the ALU which contains the results of the previous microinstruction.

Being an interrupt driven microprocessor, the BIU does not operate from a set of machine instructions (macroinstructions). It is driven by interrupt stimuli which select the beginning address of the appropriate interrupt microroutine. Referring to Figure 14, the general operation of the BIU follows. When interrupt stimuli are received by the Interrupt Control Unit, the highest priority interrupt is determined, and an interrupt request signal is sent to the Condition Code Multiplexer and an associated 4-bit vector address is sent to the Vector Mapping PROM, which fetches the starting address for the appropriate interrupt microroutine in the Microprogram PROM. When the currently executing microinstruction checks the Condition Code Multiplexer for interrupts, the interrupt is detected, the current microinstruction plus one is stored in the Microprogram Controller stack, and an interrupt service routine is executed. This routine stores, in the BIU Stack, the Interrupt Control Unit interrupt mask and status and the contents of the CPU Interrupt Vector Register (See Table IV). The last microinstruction of the interrupt service routine obtains from the Vector Mapping PROM the address of the first microinstruction of the appropriate interrupt routine. The microinstruction is fetched from the Microprogram PROM and placed in the Pipeline Register for execution. The microroutine may cause data to be obtained from the BUS Translation Unit, processed in the ALU, and sent to the I-Bus interface for processing to the PE memory.

Control Section

Interrupt Control Unit. The interrupt Control unit structure is shown in Figures 16 and 17. This unit accepts interrupts from the BTU, I-Bus Interface Section, and the Set/Clear Decoder (program generated interrupts), selects the highest priority unmasked interrupt, acknowledges the interrupt, and generates a 4-bit vector for the Vector Mapping PROM, which provides an 8-bit interrupt vector address for an interrupt subroutine in the Microprogram PROM. The interrupt request is processed by two Am2914 Vectored Priority Interrupt Encoders which are supported by an Am2913 Priority Interrupt Expander.

The Am2914 is a high-speed, 8-bit priority interrupt unit that is cascadable to handle any number of priority interrupt request levels. Two Am2914's are used to provide 16 levels of interrupts.

Each of the Am2914's receives interrupt requests on eight interrupt input lines, $P_0 - P_7$. A low level is a request. An internal latch is

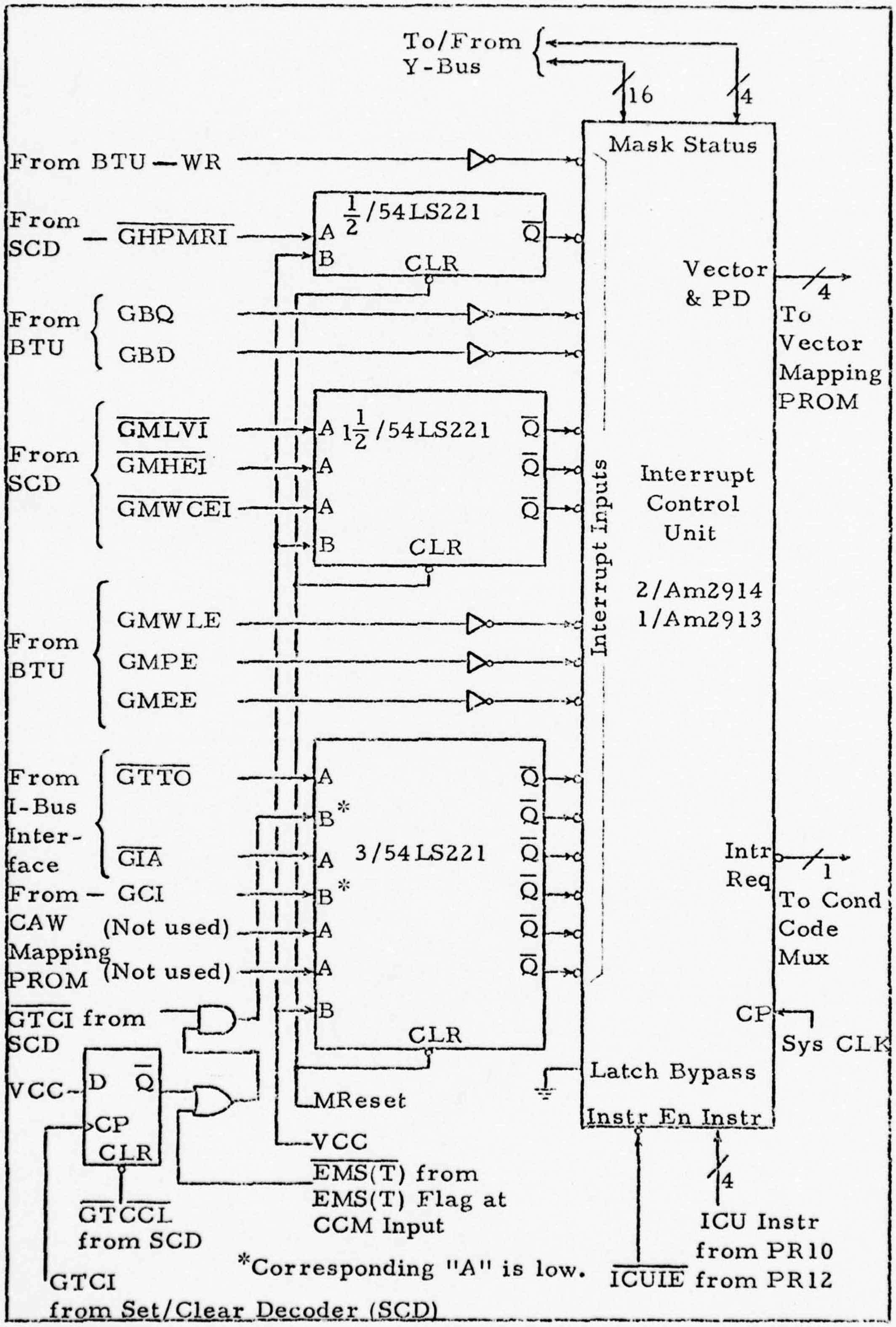


Fig. 16. Interrupt Control Unit and Associated Input Circuitry

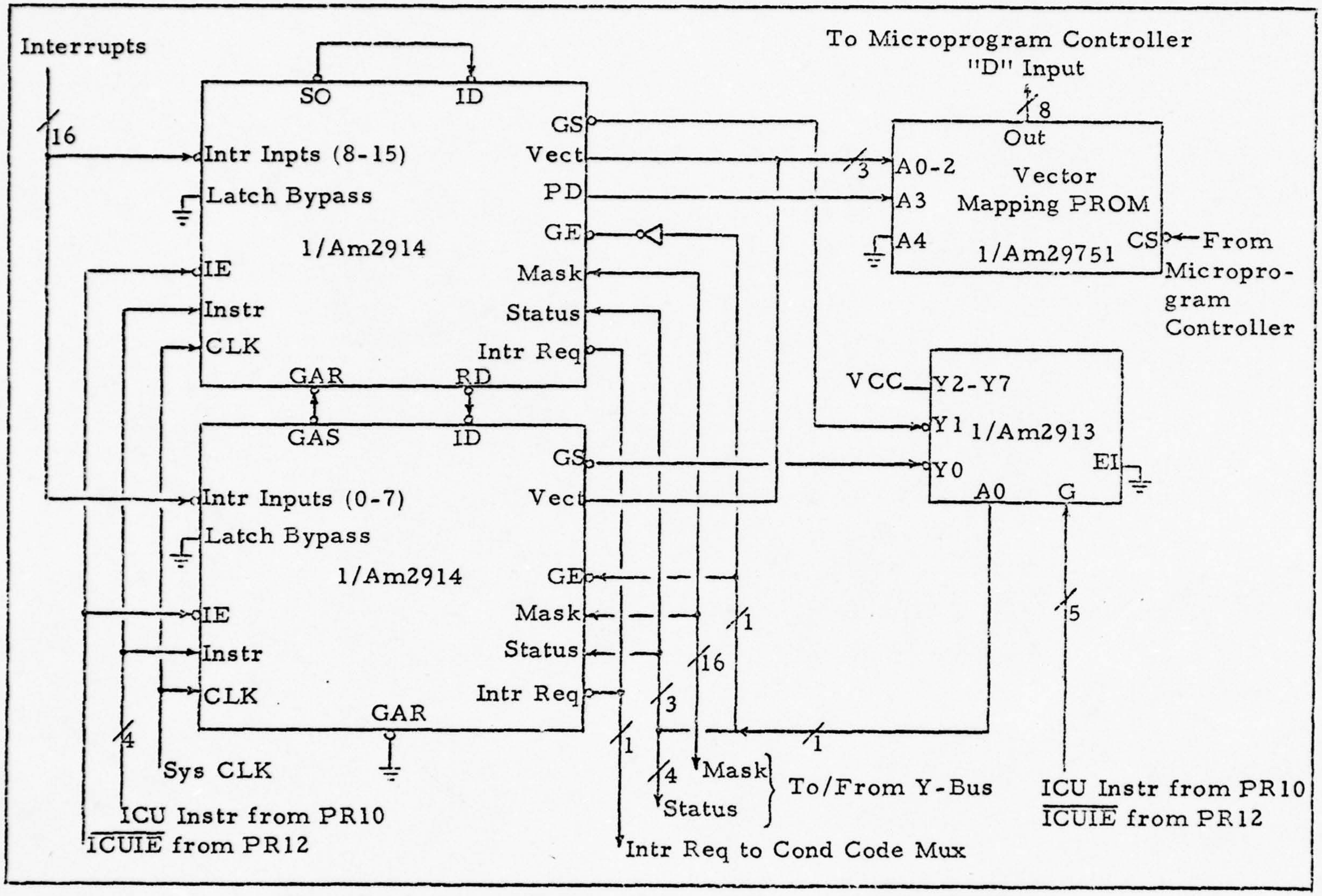


Fig. 17. Interrupt Control Unit and Vector Mapping PROM

available at the input to catch low pulese on the lines. An 8-bit mask register is used to mask individual interrupts. Requests in the interrupt register are ANDed with the corresponding bits in the mask register and the results are sent to an internal 8-input priority encoder, which determines the highest priority nonmasked interrupt input and forms a binary coded interrupt vector. (In the 16 interrupt level configuration, the Parallel Disable (PD) output of the higher priority group serves as the high order vector bit.) An internal status register is used to point to the lowest priority at which an interrupt will be accepted. The contents of the status register are compared with the output of the priority encoder, and an interrupt request occurs if the vector is greater than or equal to the status. When a vector is read from the Am2914, the status register is automatically updated to point to one level higher than the vector read (Ref 5:2-3).

The am2913 Priority Interrupt Expander, an 8-line-to3-line priority encoder, is used to encode the high order bit of the status when the status register is read. The Am2913 is gated by the Am2914 instruction lines so that its output is enabled only during a Read Status instruction. An inverter at the input to the Group Enable (GE) decodes the high order status bit during loading of the status register.

The Am2914 is controlled by a 4-bit instruction field, I_0-I_3, which allows 16 instructions (Table XII, Chapter IV), associated with interrupt control, vector read, mask register control, and status register control. The command on the instruction line is executed if the Instruction Enable input (IE) is low and ignored if IE is high.

As shown in Figure 16, there are 14 interrupt signals to the Interrupt Control Unit. They are the 11 CPU interrupts, which were discussed in Chapter II (the Global Message Received interrupt is not shown because it is handled in software and not processed through the Interrupt Control Unit), and three BIU interrupts, the Word Received (WR) interrupt, the Global Interrupt Acknowledge (GIA) interrupt, and the Global CAW (GCI) interrupt. The Word Received interrupt is generated when the Input Data Bit Counter in the BTU counts the 17th bit of an incoming word, designating that a word has been assembled in the BTU's Input Shift Register. The Global Interrupt Acknowledge interrupt is generated by the I-Bus Interface when the PE CPU acknowledges an interrupt request. The Global CAW interrupt is generated by the CAW Mapping PROM when a CAW for the BIU is identified.

The Word Received interrupt and five of the 11 CPU interrupts are received from the BTU. The interrupts are in the form of high-level signals, and they require a clear signal after being serviced. The interrupt signals are fed through inverters to the Am2914 "P" inputs. When these interrupts are serviced by their respective interrupt routines, they are cleared in the Am2914 with the Clear Interrupt, Last Vector Read (CILVR) instruction and they are cleared in the BTU with microprogram generated clear interrupt signals.

The other six CPU interrupts are software generated and are processed through the Interrupt Control Unit only when they are being masked out by the PE CPU (see discussion on the arm/disarm interrupts in the Interrupt Interface/Control Function Section of Chapter II). These interrupts and the two interrupts from the I-Bus Interface are in the form of low-level signals that trigger monostable multivibrators (54LS221) which generate low-level pulses to the Am2914 "P" inputs. The reason for using pulses is to eliminate the need to issue clear interrupt signals when servicing the interrupts. All that is required to clear an interrupt is to issue the Clear Interrupt, Last Vector Read instruction to the Am2914.

The Global Transmission Completed interrupt (GTCI) is a software generated interrupt that occurs under two conditions (refer to the Global Activate Output and the Global Transmission Completed routines in Appendix B). The first condition occurs when the last message word is placed in the Output Data Buffer and the message is not transmitted because the allocated time-slot has not yet arrived. Under this condition a GTCI signal is issued which sets a D-type flip-flop (Figure 16). The $\overline{Q}$ output of the flip-flop is ORed with an $\overline{\text{EMS(T)}}$ signal, which is issued when the message is transmitted. When the $\overline{\text{EMS(T)}}$ is issued, a monostable multivibrator at the Am2914 input is tirggered. The second condition occurs when message transmission is completed and the Global Transmission Completed interrupt is masked out by the PE CPU. Under this condition, a $\overline{\text{GTCI}}$ signal is issued to the multivibrator. When the mask is cleared, the interrupt is detected.

Vector Mapping PROM. One Am29751 32-word by 8-bit PROM with three-state outputs is used to map the Interrupt Control Unit vector outputs to the "D" inputs of the Am2910 Microprogram Controller. The interrupt vector address is used as the starting microprogram PROM address of a microprogram interrupt service routine.

Microprogram Controller. One Am2910 Microprogram Controller (Figure 15) is used to select the source of the address of the next microinstruction to be executed. During each microinstruction cycle, the Microprogram Controller provides an 8-bit address to the Microprogram PROM from one of four sources: (1) a microprogram counter (PC), which usually contains an address one greater than the previous address; (2) an external "D" (direct) input; (3) a register/counter (R) retaining data loaded during a previous microinstruction; or (4) a five-deep last-in, first-out stack (MC STK). (The Am2910 actually accepts a 12-bit address at the "D" input and outputs a 12-bit address at the "Y" output. Because only an 8-bit address is required in this design, the four most significant address bits are made permanently low.)

The Microprogram Controller generates three enable signals, $\overline{\text{PL}}$, $\overline{\text{VECT}}$, and $\overline{\text{MAP}}$, to allow three sources to be fed into the "D" input. In this design, $\overline{\text{PL}}$ selects the next Address Field of the Pipeline Register as the direct input source; $\overline{\text{VECT}}$ selects the Vector Mapping PROM as the direct input source; and $\overline{\text{MAP}}$ selects the CAW PROM register as the direct input source. For each Microprogram Controller instruction, one and only one of the three enable signals is low.

The Microprogram Controller Stack is used to provide return address linkage when executing microsubroutines or loops. Anytime the stack is full, a $\overline{\text{FULL}}$ warning output occurs. When additional information is pushed onto a full stack, the information at the top of the stack is overwritten and lost. The $\overline{\text{FULL}}$ warning output is used to set a warning light on the PE control panel (not part of this design). The warning output could have been used to set an interrupt, but this was not done because this condition has not yet been considered in the software developed by AFAL. If it is desirable to make this an interrupt condition, the $\overline{\text{FULL}}$ signal can be sent to one of the two unused inputs in the Interrupt Control Unit.

The Am2910 is controlled by a 4-bit instruction field, I_0 - I_3, which selects the address of the next microinstruction to be executed (Table V, Chapter IV). Four of the instructions are unconditional. Ten of the instructions have operations which are partially controlled by an external,data-dependent condition. Three of the instructions have operations which are partially controlled by the contents of the internal register/counter.

In the ten conditional instructions, the data-dependent condition is applied to the Condition Code Input ($\overline{CC}$). If the $\overline{CC}$ input is low the test is considered to have passed, and the action specified in the instruction occurs; otherwise, the test has failed and the next sequential microinstruction occurs. Testing of the $\overline{CC}$ input is enabled when the Condition Code Enable ($\overline{CCEN}$) input is low. In this design $\overline{CCEN}$ is connected to ground. Conditional instructions are executed unconditionally by selecting a Condition TRUE input, which is connected to ground. The signal to the $\overline{CC}$ input is obtained from the Condition Code Multiplexer.

Condition Code Multiplexer. The Condition Code Multiplexer is a 32-line-to-1-line Multiplexer composed of two 54150 16-input multiplexers (Figure 18). A 5-bit microinstruction field is used to select one of the 32 inputs (five are not used at this time). The first "D" input (D0) is connected to VCC and is the TRUE condition. Ten other signals are directly connected to the "D" inputs; one signal from the Interrupt Control Unit, three signals from the Status Register, three signals from the I-Bus Master Control Logic, two signals from the I-Bus Interrupt Control Logic, and one signal from the CPU Interrupt Vector Buffer.

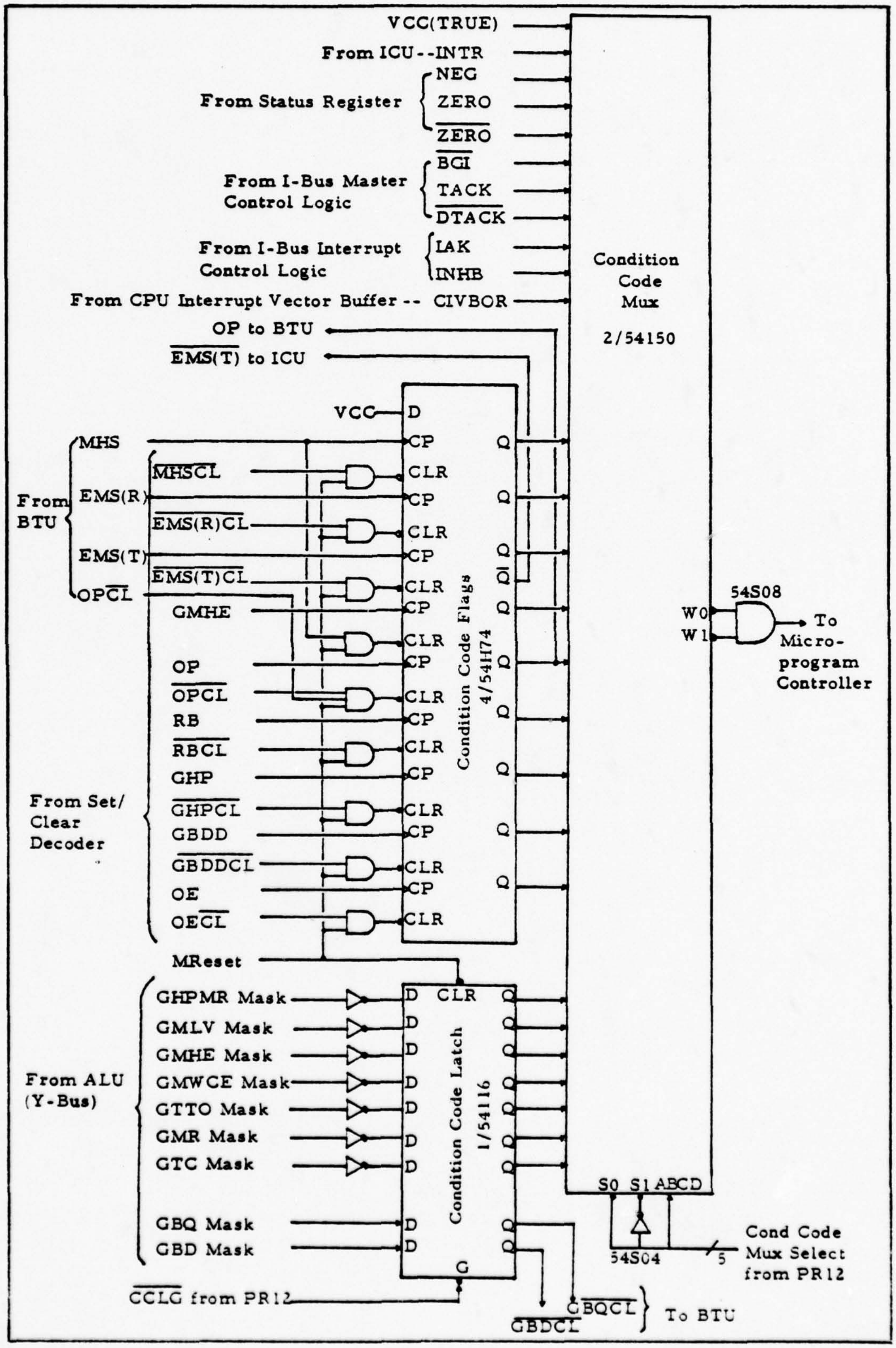


Fig. 18. Condition Code Multiplexer

Nine signals are sent to the "D" inputs through the Condition Code Flags, five 54H74 dual D-type positive-edge-triggered flip-flops with preset and clear. Three signals from the BTU and six signals from the Set/ Clear Decoder. Seven other signals to the Condition Code "D" inputs are from the Condition Code Latch, one 54116 dual 4-bit latch with clear, which latches seven bits of data from the ALU output.

CAW Mapping PROM. The CAW Mapping PROM, which is used to decode CAW's from the PE CPU, consists of two Am29760 256-word by 4-bit PROM's with open collector outputs (Figure 19). When a CAW is recognized as being for the BIU, an 8-bit starting address for a servicing microroutine is sent from the CAW Mapping PROM to the CAW Mapping PROM Register for transfer to the Microprogram Controller "D" input. The CAW Mapping PROM output is also sent to an 8-input NAND gate. When a CAW is identified, the output of the NAND gate goes high triggering a pulse generator at the Interrupt Control Unit input indicating a CAW interrupt.

As shown in Table II, the BIU is presently required to execute six CAW's, words with decimal values three, five, seven, nine, 11, and 13. In this design the seven least significant bits (LSB) of the CAW are decoded, allowing for 128 CAW's with decimal values between zero and 127. The seven bits are connected to the seven least significant bits of the address (A) inputs to the CAW Mapping PROM. The most significant address bit is connected to the I-Bus DRCV line. As seen in Table II, the same CAW can require either a write or read operation. The lower half of the CAW Mapping PROM is used for CAW's involving CPU Write and the upper half is used for CAW's involving CPU Read. Address locations

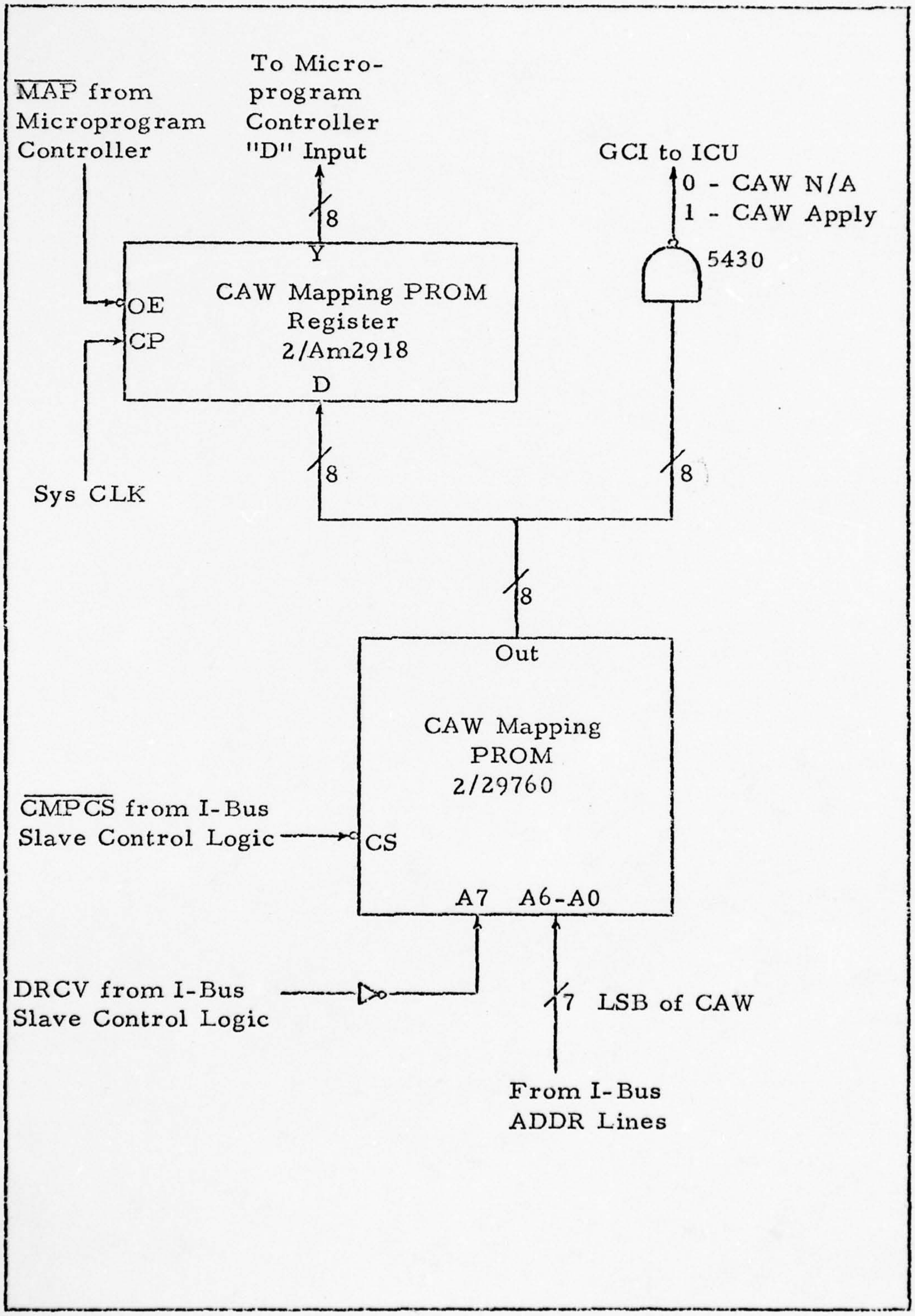


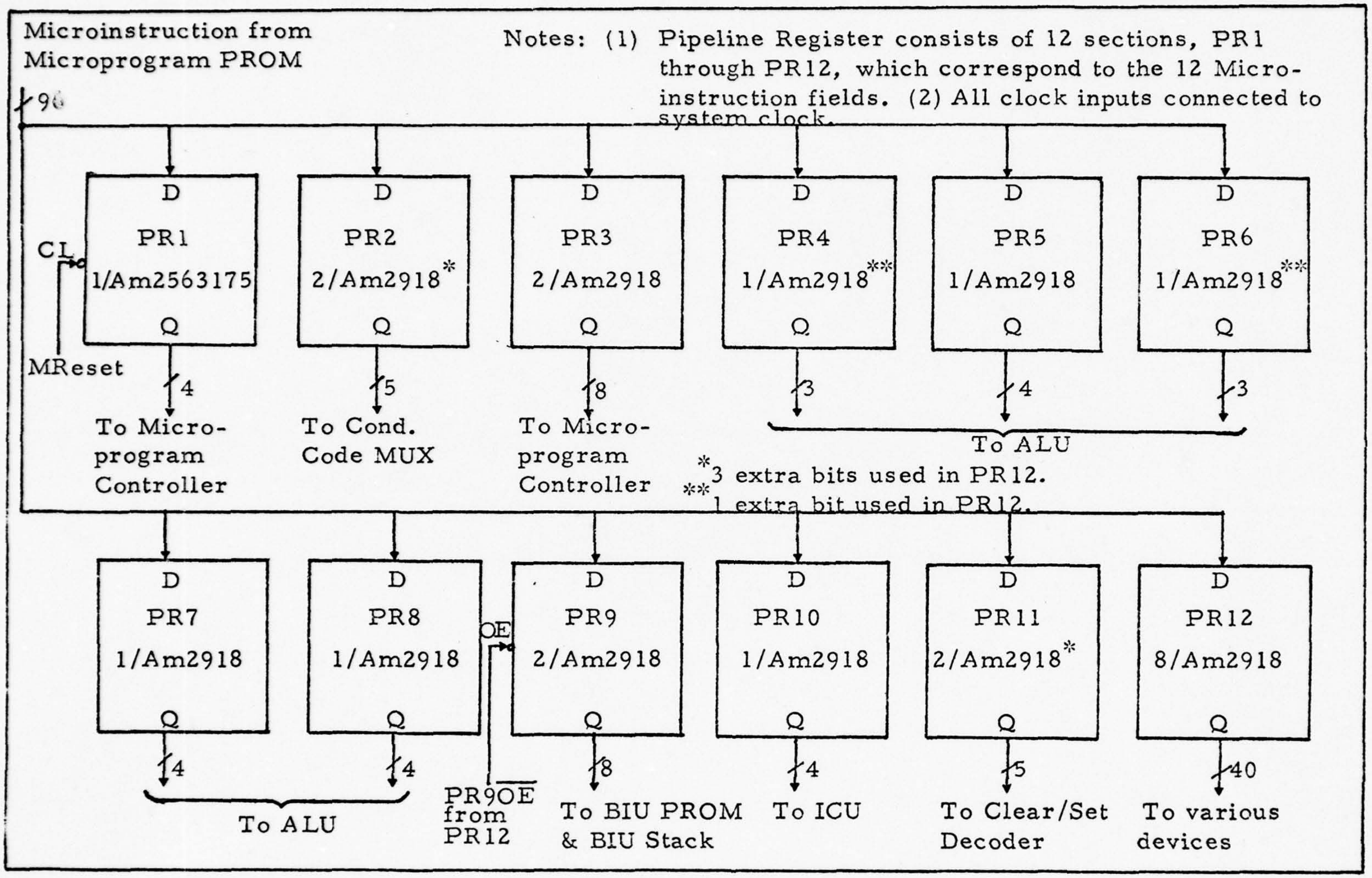
Fig. 19. CAW Mapping PROM

not corresponding to an applicable CAW are loaded with all ones. The PROM's Chip Select ($\overline{CS}$) signal is obtained from the I-Bus Slave Control Logic.

Microprogram PROM. The Microprogram PROM consists of 23 Am29761 256-word by 4-bit PROM's with three state outputs. The configuration used provides a 256-word by 92-bit memory which contains the microinstructions required for operating the BIU. The addresses for the microinstructions are provided by the Microprogram Controller, and the fetched microinstructions are placed into the Pipeline Register.

Pipeline Register. Twenty two Am2918 quad D registers with standard and three-state outputs and one Am25LS175 quad D register with common clear make up the 92-bit Pipeline Register (Figure 20). The standard output is used in all the Am2918's except for two, which use the three-state output. These two Am2918's are one of two sources for the BIU PROM address. The Am25LS175 is used for the Next Address Control Instruction Field of the microinstruction. Its use is required for system initialization. During the power-up sequence, a Master Reset signal is issued by the PE CPU. This signal is used to clear the Am25LS175 which then presents the Zero instruction to the Am2910 Microprogram Controller. This instruction forces the microprogram location counter to address Zero in the Microprogram PROM which contains the power-up microinstruction.

The microinstruction format employed in this design consists of 12 fields. These fields are discussed in detail in Chapter IV. With the exception of two fields, no attempt was made to overlap fields or



Notes: (1) Pipeline Register consists of 12 sections, PR1 through PR12, which correspond to the 12 Microinstruction fields. (2) All clock inputs connected to system clock.

*3 extra bits used in PR12.

**1 extra bit used in PR12.

Fig. 20. Pipeline Register

encode control signals to reduce the microinstruction length. Since the design is for a laboratory development model, it was felt that operational speed and system flexibility was more important than a short microinstruction. Reformatting of the microinstruction would be appropriate after the BIU is operating and all timing requirements satisfied.

Set/Clear Decoder. As mentioned above, one microinstruction field does encode a group of control signals. This field is the 5-bit Set/Clear field which encodes 32 control signals for the BTU, Condition Code Flags, Interrupt Control Unit, and the I-Bus Interface. The Set/Clear Decoder which decodes this field is composed of four Am25LS151 8-input multiplexers (Figure 21).

BIU Stack. The BIU Stack is required when servicing an interrupt to save the current Interrupt Control Unit mask and status and to save the current contents of the CPU Interrupt Vector Register (see Table IV). The stack is provided by four Am2930 Program Control Units. These units contain a 17-word by 4-bit last-in-first-out stack (plus other functions not used in this design), which operates with the instructions shown in Table XI.

Data Manipulation Section

Arithmetic Logic Unit. The Arithmetic Logic Unit consists of four Am2901's and one Am2902 (Figures 22 and 23). The unit is actually a 16-bit microprocessor. Each Am2901 is a 4-bit microprocessor slice consisting of a 16-word by a 4-bit 2-port RAM (providing 16 general purpose register), a Q Register, a high-speed arithmetic logic unit, and associated shifting, decoding, and multiplexing circuitry. The Am2901

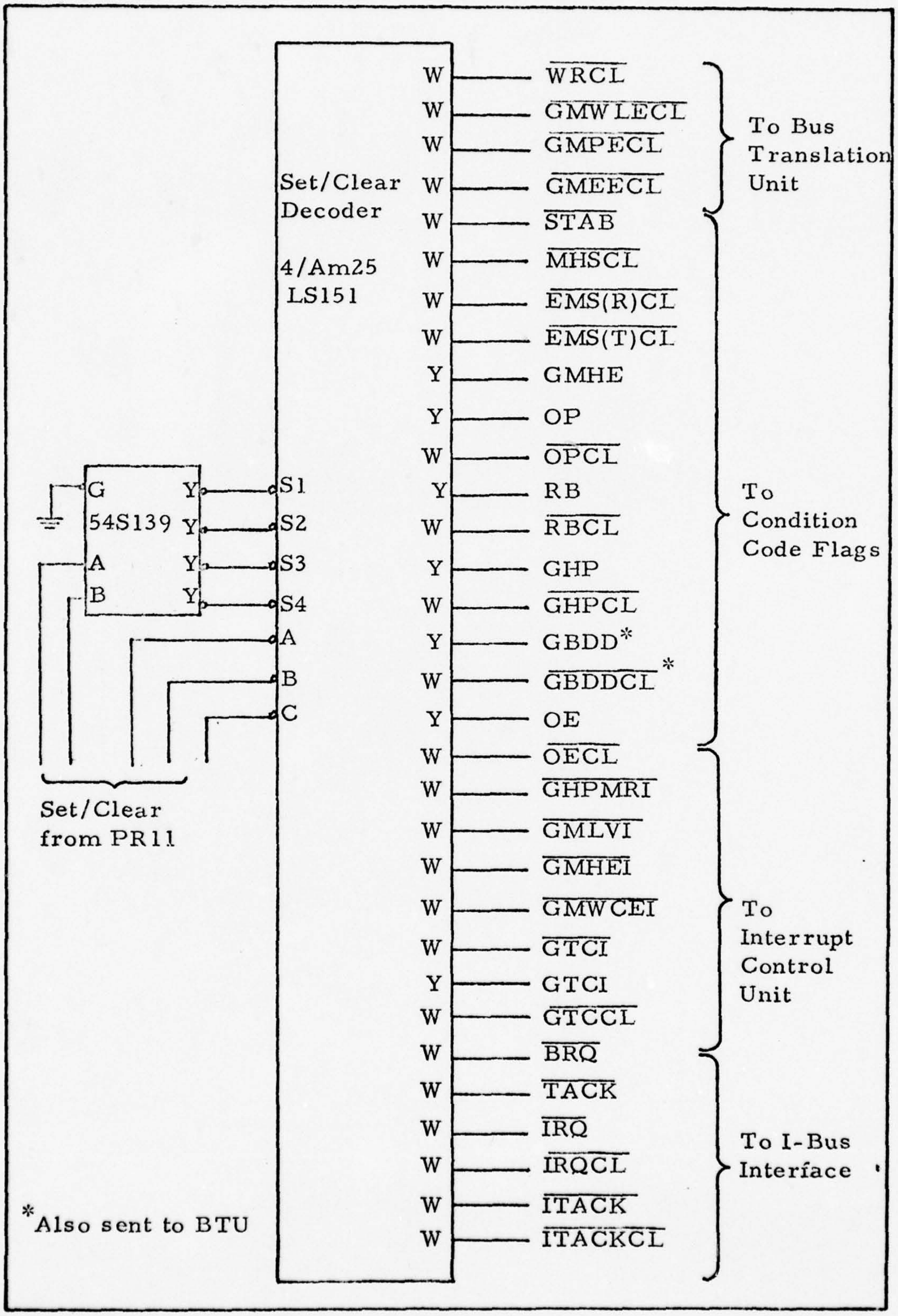


Fig. 21. Set/Clear Decoder

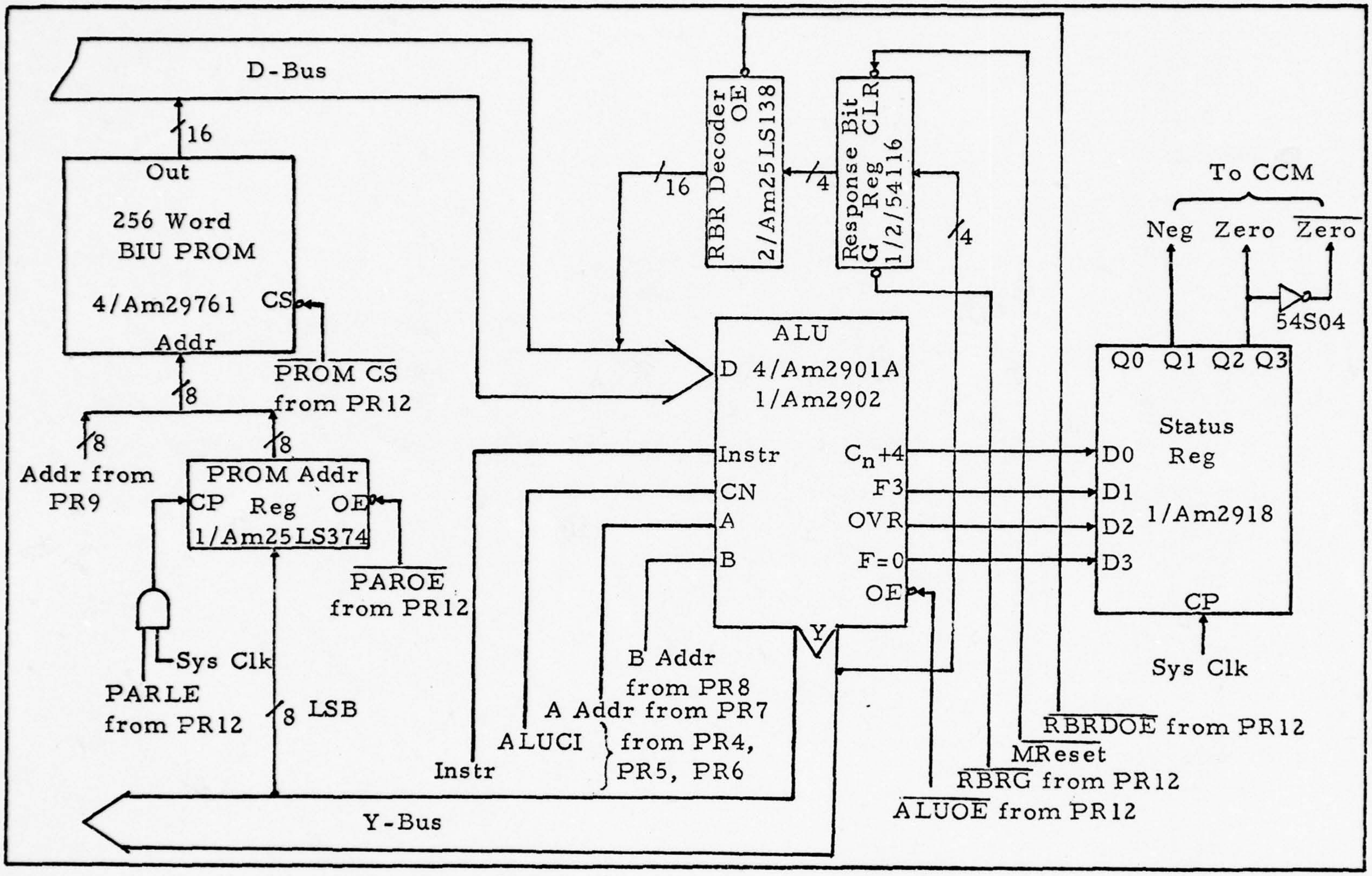


Fig. 22. Data Manipulation Section (Excluding BIU Stack)

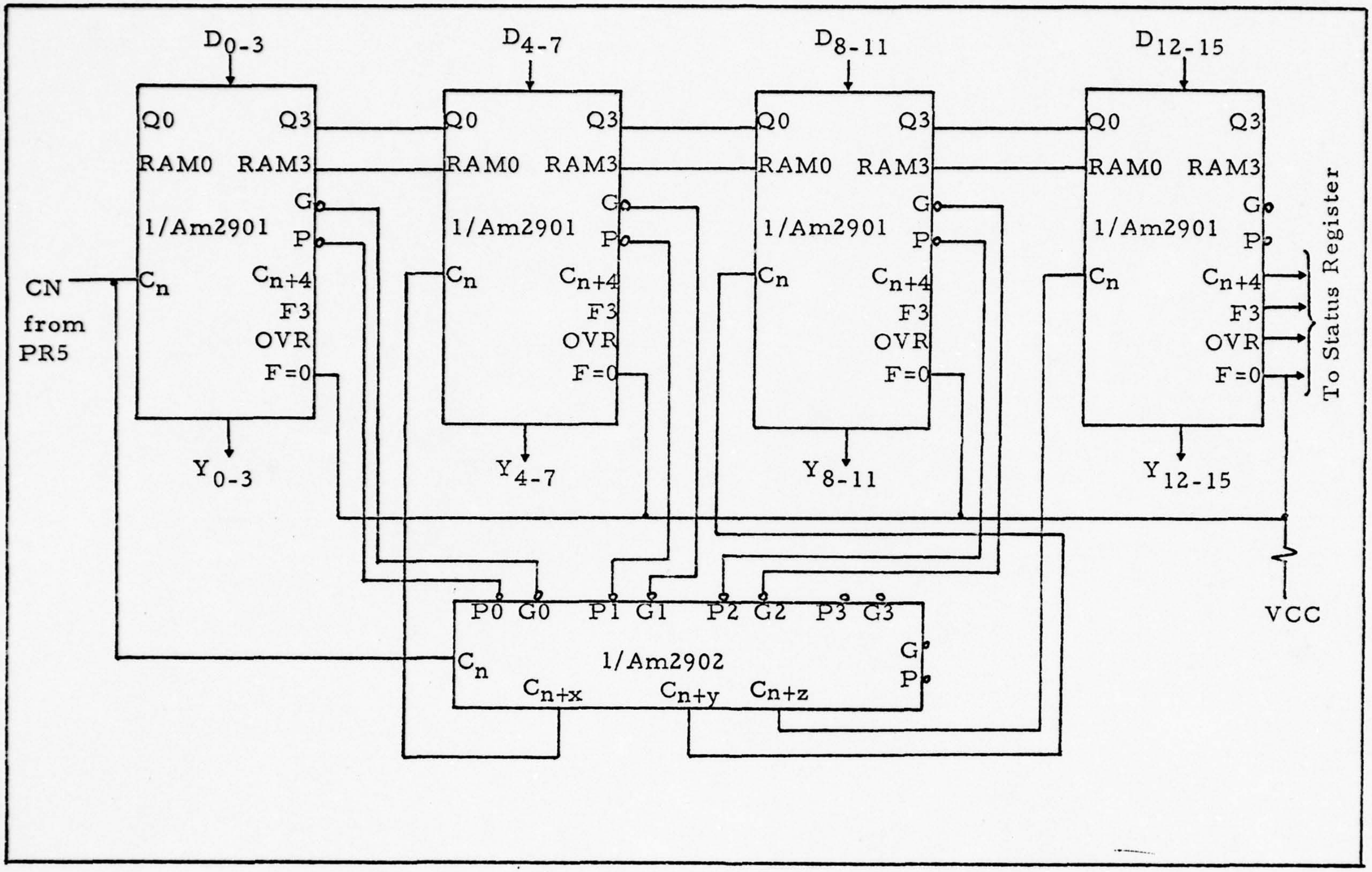


Fig. 23. Arithmetic Logic Unit (From Ref 4:18)

is controlled by a 9-bit microinstruction field organized into three groups of three bits each (Tables VII, VIII, X). These groups select the ALU source oeprands, the ALU functions, and the ALU destination register. The Am2902 high-speed look-ahead carry generator is used to provide faster ALU operation. It accepts the propagate and generate signals from each 4-bit microprocessor slice and provides the carry-in singal to the next stage and an anticipatory signal to successive stages.

The BIU registers required for processing messages are located in the 2-port RAM. These registers are listed in Table IV. Data in the RAM is read from an A-port of the RAM as controlled by a 4-bit "A" address field input and by a B-port of the RAM as controlled by a 4-bit "B" address field input. New data is written into the RAM by the "B" address field.

TABLE IV

Registers in ALU RAM

Mnemonic	Register
ACC	Accumulator
ALU(CISR)	CPU Interrupt Status Register
ALU(CIVR)	CPU Interrupt Vector Register
ALU(ILR)	Input Length Register
ALU(IAR)	Input Address Register
ALU(IDR)	Input Data Register
ALU(OLR)	Output Length Register
ALU(OAR)	Output Address Register

The ALU data output is routed to several destinations. The data can be sent out of the Am2901 (three-state output) and it can also be stored in the RAM or Q Register. As seen in Table X, eight possible combinations of ALU destination functions are available.

The ALU has four status-oriented outputs. These are carry-out (C_{n+4}), most significant bit or sign (F_3), overflow (OVR), and zero detect ($F = 0$). These status outputs are sent to the Status Register.

Status Register. The Status Register consists of one Am2918 quad D register with standard and three-state outputs. This register is required for the one level pipelining which is used in this design. It presents the results (status) of the previous microinstruction operation to the Microprogram Controller.

BIU PROM and PROM Address Register. The BIU PROM which consists of four Am29761 256-word by 4-bit PROM's with three-state outputs, is used to store the constants and masks required for processing messages and to store the CPU interrupt vector words and Message Identity Associative Match Map words (Table XI). As shown in Fiugre 22, there are two sources for the 8-bit address inputs. The PROM Address Register, which consists of one Am25LS374 8-bit register with three-state outputs, is used when addressing Message Identity Associative Match Map words. The addresses for these words are calculated in the ALU. The Pipeline Register PR9 is used to address all other words.

Response Bit Register and Decoder. As discussed in Chapter II, during message reception, the response bit of a particular Message Identity Associative Match Map word is determined by the binary value of the four least significant bits of the Message Identity Field of the

Message Header word. In this design, these four bits are extracted from the Message Header word and placed in the Response Bit Register, one-half of a 54116 dual 4-bit latches with clear. The response bit is determined from these four bits by the Response Bit Register Decoder, which consists of two Am25LS138 3-line to 8-line decoder/demultiplexers. The output of the decoder is sent to the ALU for comparison against the Message Identity Associative Match Map word (Figure 22).

I-Bus Interface Section

I-Bus Registers. There are six registers associated with this section. Four registers that interface directly with the I-Bus are the BMID Register, I-Bus Flag Register, I-Bus Address Register, and the I-Bus Data Register. The other two registers of the section are the Input Queue Pointer Register and the CPU Interrupt Vector Buffer, a first-in first-out memory.

The BMID Register (Figure 24) is used to place the BIU's 4-bit bus master identification (BMID) code onto the I-Bus. The register consists of one Am2918 quad D register with standard and three-state outputs. Only the three-state outputs are used. The output enable signal is obtained from the BMID Register Flag which is set by a $\overline{\text{BGI}}$ signal from the Bus Request Flag in the I-Bus Master Control Logic. The inputs to the BMID Register are hardwired to provide the appropriate BMID code.

The I-Bus Flag Register (Figure 24) is used to place the appropriate TRQ, IOSL, and DRCV signals onto the I-Bus. The signals are received at the register inputs from the BIU PROM via the D-Bus. The I-Bus Flag Register consists of one Am2907 quad open-collector bus

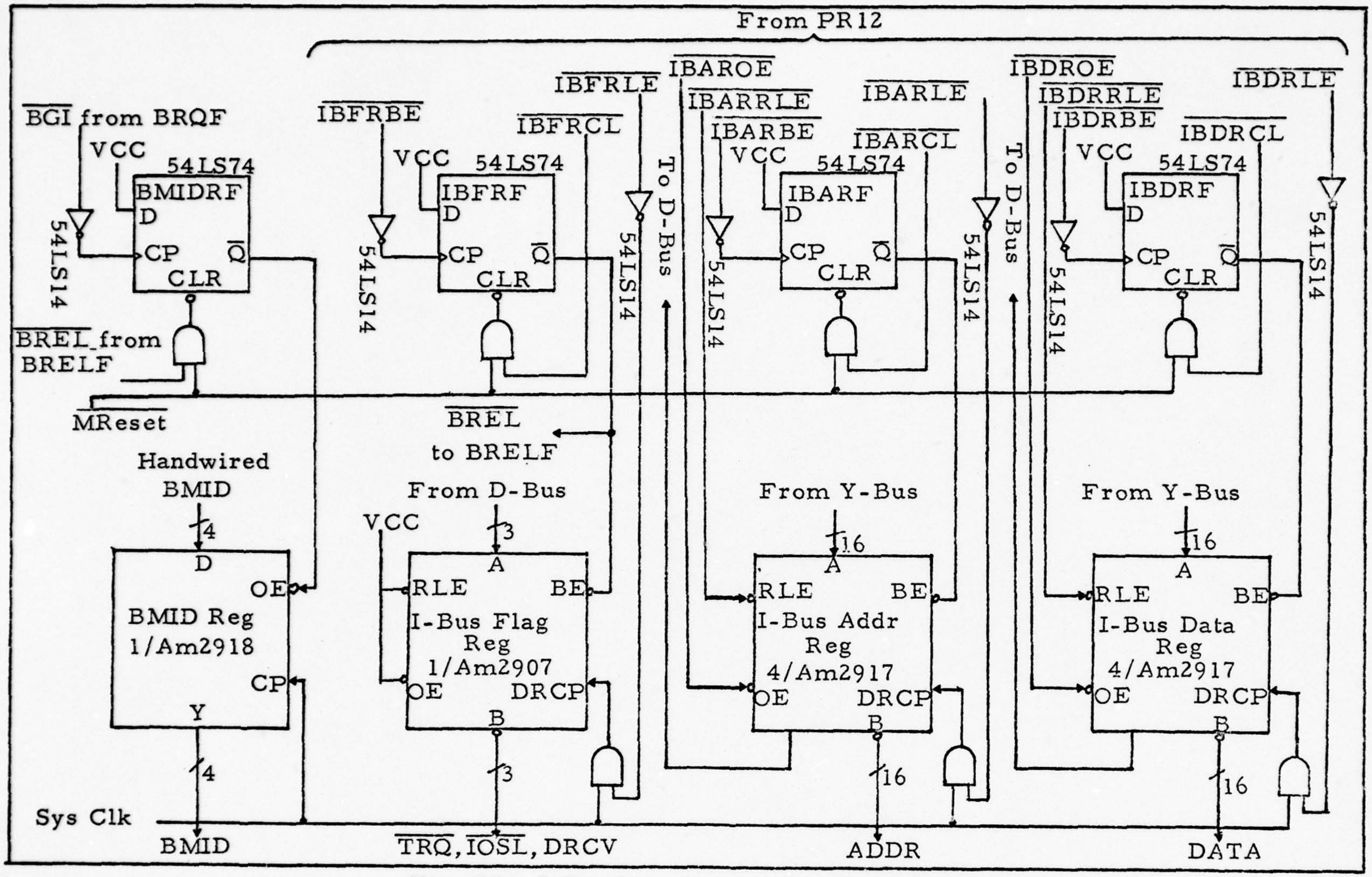


Fig. 24. I-Bus Interface--I-Bus Registers

transceiver with three state receiver and parity (parity not used). The bus enable signal is obtained from the I-Bus Flag Register Flag which is set by an $\overline{\text{IBFRBE}}$ signal from a microinstruction (PR12).

The I-Bus Address Register and I-Bus Data Register (Figure 24) are used, respectively, to place address and data information onto the I-bus. Each of these registers consist of four Am2917 quad three-state bus transceivers with interface logic. Data is entered into the "A" input from the BIU Y-Bus on the low-to-high transition of the System Clock when a $\overline{\text{LE}}$ signal is issued by the microinstruction. The data is transmitted onto the I-Bus when a $\overline{\text{BE}}$ signal is received from the respective register flag which is set by a $\overline{\text{BE}}$ signal from the microinstruction. Data from the I-Bus is received into the registers when an $\overline{\text{RLE}}$ signal is issued by the microinstruction. The data is then sent onto the BIU D-Bus when the microinstruction issues an $\overline{\text{OE}}$ signal.

As seen in Fiugre 25, the Input Queue Pointer Register is a 16-bit register consisting of four Am2918 quad D registers with standard and three-state outputs. Inputs to the 13 most significant bits of the register are hardwired to a set value. The inputs to the three least significant bits are connected to the three least significant bit outputs of an Am25LS161 synchronous 4-bit binary counter with asynchronous clear. The counter is being used as a modulo-eight counter. (Note that in the DP/M system, bit-0 is the most significant bit and in the AMD devices, bit-0 (or A) is the least significant bit.)

The CPU Interrupt Vector Buffer is used to temporarily store a CPU interrupt vector address until the PE CPU acknowledges the interrupt request. Two Am2812A 32-word by 4-bit first-in first-out (FIFO)

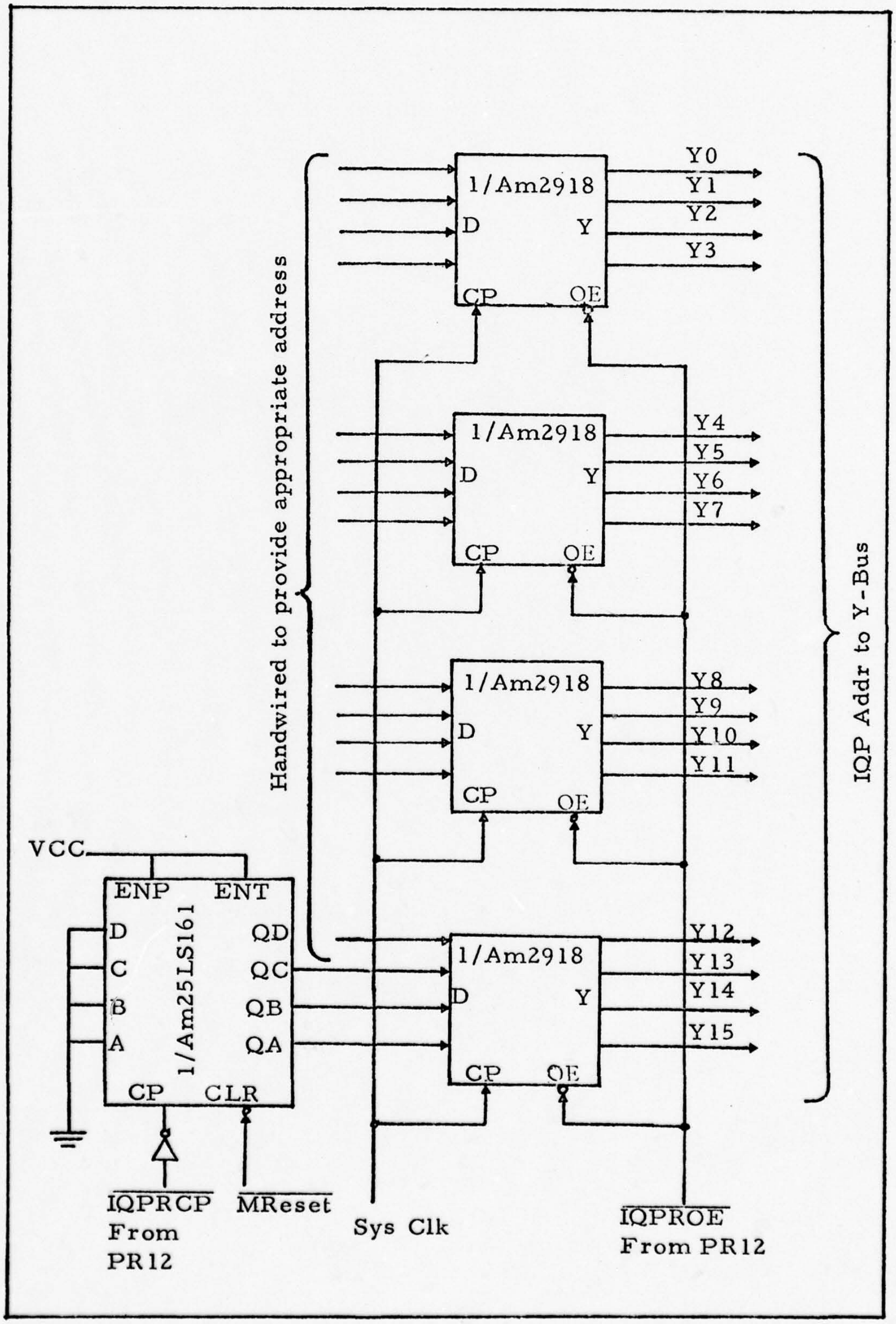


Fig. 25. Input Queue Pointer Register

memories are used for the buffer (Figure 26). The use of a FIFO memory permits more than one interrupt vector address to be held while awaiting PE CPU acknowledgement, which may take a while. The PE CPU does not check for an interrupt request until it completes executing the current macroinstruction, and then it must gain control of the I-Bus for which it has the lowest bus access priority.

Data from the Y-Bus is written into the CPU Interrupt Vector Buffer when CIVBPL is issued by the microinstruction. Data is read from the buffer when CIVBPD is issued by the microinstruction. An output ready signal (CIVBOR),which indicates that data is available at the output, is connected to the Condition Code Multiplexer. After an interrupt vector is read from the buffer, the output ready signal is checked to see if another interrupt request to the PE CPU is required. A Flag signal indicates when the buffer contains more than 15 words. Since the stacking of 15 interrupts is not expected, the Flag signal is used to set a warning light on the PE control panel (not part of this design).

I-Bus Master Control Logic. During a direct memory access operation (refer to the Direct Memory Access Function paragraph in Chapter II for a description), the BIU acts as a master device on the I-Bus. The I-Bus Master Control Logic (Figure 27) is used to perform this function. There are four flags, 54LS74 D-type flip-flops, associated with this circuitry. They are the Bus Request (BRQ) Flag, Bus Release (BREL) Flag, Transfer Acknowledge (TACK) Flag, and the Delayed Transfer Acknowledge Flag. The Bus Request and Bus Release Flags are used for bus control. The outputs of these two flags are sent through buffers with open collector outputs to provide wired-OR signals as specified in Reference 3.

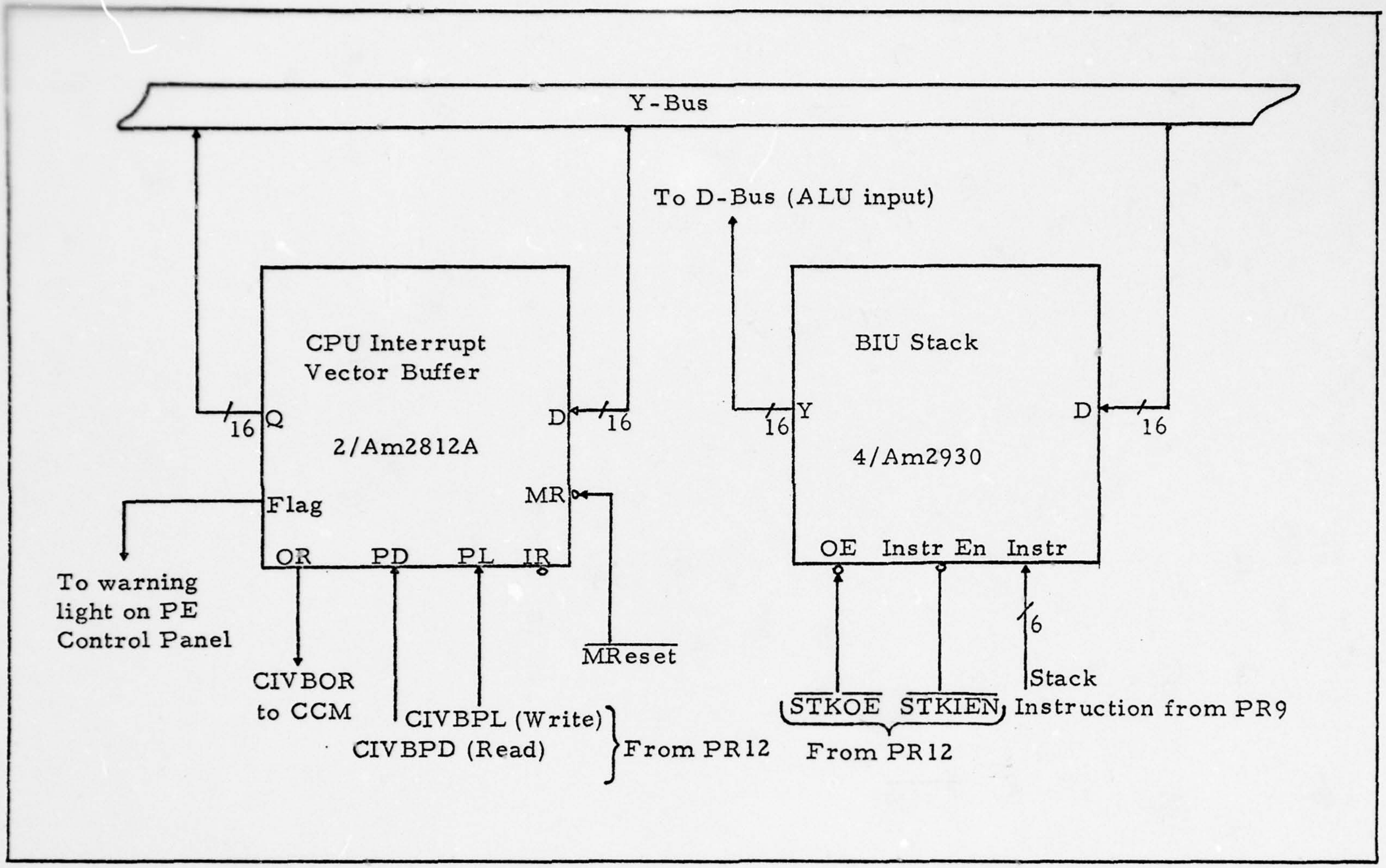
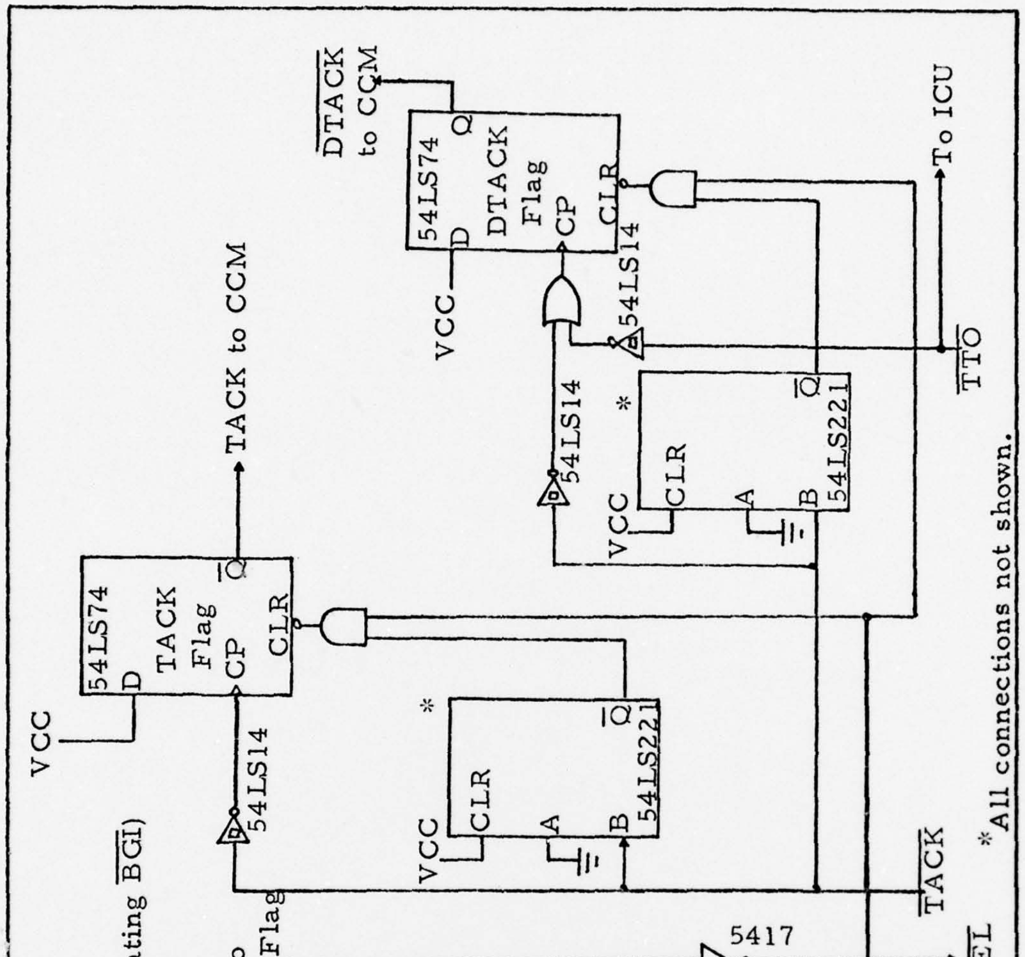


Fig. 26. CPU Interrupt Vector Buffer and BIU Stack



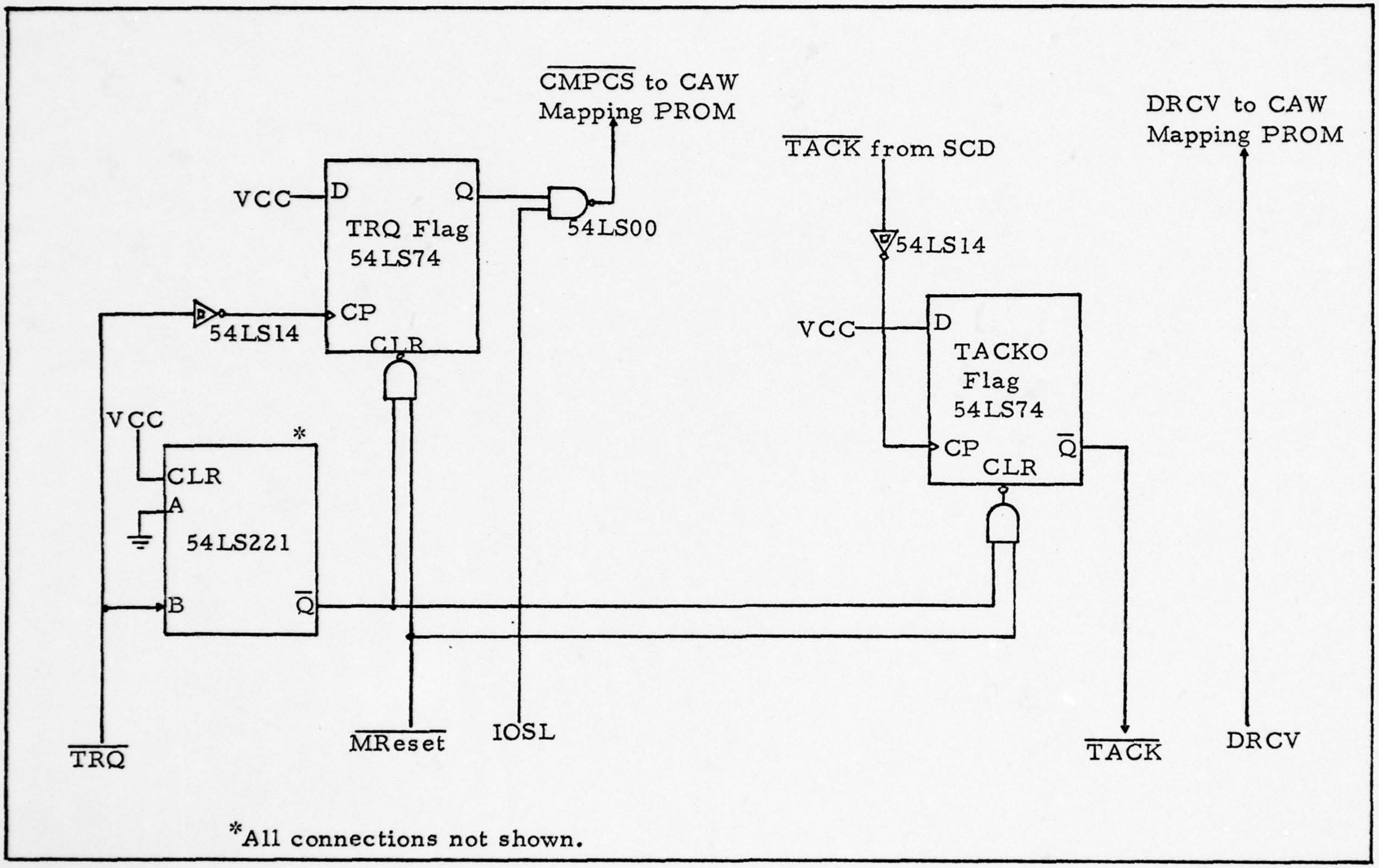
*All connections not shown.

Fig. 27. I-Bus Interface--I-Bus Master Control Logic

The TACK Flag indicates whether the TACK line is high or not. After obtaining control of the bus, the BIU must wait until any previous data transfer is completed (TACK line high) before it starts its data transfer operation. The flag is cleared by either the Master Reset (MReset) signal from the PE CPU or by a low-level pulse generated by a monostable multivibrator when the TACK line goes from low to high.

The Delayed TACK Flag indicates transfer acknowledgement from the PE memory (slave), and is set by a $\overline{\text{TACK}}$ signal from the memory. The Delayed TACK Flag can also be set by a Transfer Time Out signal which is sent by the PE memory when a nonexistent memory location is addressed.

I-Bus Slave Control Logic. The I-Bus Slave Control Logic (Figure 28) is used when the BIU responds to Command Address Words. During this operation (refer to the Program Command Response Function paragraph in Chapter II for description), the BIU acts as a slave device on the I-Bus. There are two flags, 54LS74 D-type flip-flops, associated with this circuitry, the Transfer Request (TRQ) Flag and the Transfer Acknowledge Out (TACKO) Flag. When the TRQ Flag is set by a $\overline{\text{TRQ}}$ signal from the I-Bus and the I-Bus IOSL line is high, a chip select signal ($\overline{\text{CMPCS}}$) is generated and sent to the CAW Mapping PROM. If the Command Address Word applies to the BIU, the appropriate action is initiated and the TACKO Flag is set, which places a transfer acknowledge signal ($\overline{\text{TACK}}$) onto the I-Bus to the PE CPU. Both the TRQ and TACKO Flags are cleared by either the Master Reset signal from the PE CPU or by a low-level pulse generated by a monostable multivibrator when the TRQ line goes from low to high. The DRCV signal is sent directly to the most significant bit address input of the CAW Mapping PROM.



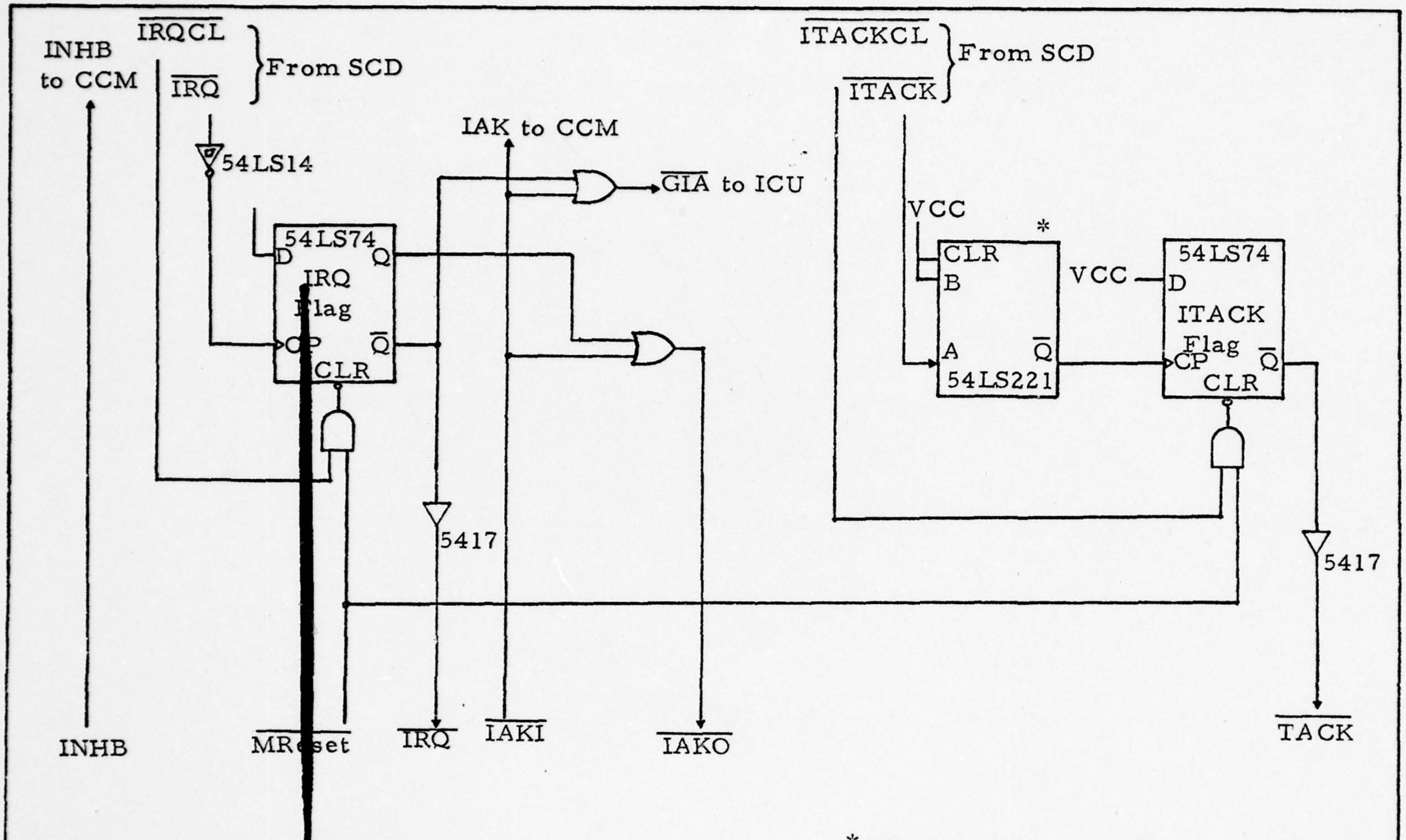
*All connections not shown.

Fig. 28. I-Bus Interface--I-Bus Slave Control Logic

CPU Interrupt Control Logic. The CPU Interrupt Control Logic (Figure 29) is used to process interrupts to the PE CPU (refer to the Interrupt Interface/Control Function paragraph in Chapter II for the interrupt operation description). After checking to see that there is no interrupt inhibit signal (INHB signal sent directly to the Condition Code Multiplexer), the BIU makes an interrupt request on the I-Bus by setting the Interrupt Request (IRQ) Flag, 54LS74 D-type flip-flop. As stated earlier, an interrupt request is not immediately acknowledged by the PE CPU. Therefore, in this design, the BIU continues on with its other operations after setting the IRQ Flag. When an interrupt acknowledge signal ($\overline{\mathrm{IAKI}}$) is received from the PE CPU, a GIA signal is sent to the Interrupt Control Unit which sets a GIA Interrupt (an interrupt internal to the BIU). When this GIA Interrupt is detected by the BIU, it completes processing the CPU interrupt. After the interrupt trap vector address is placed on the I-Bus Data lines, the Interrupt Transfer Acknowledge (ITACK) Flag is set, which issues a $\overline{\mathrm{TACK}}$ signal onto the I-Bus to the PE CPU. A monostable multivibrator is used at the input of the ITACK Flag to provide a delay of the $\overline{\mathrm{TACK}}$ signal as specified in Reference 3. The flag is set on the rising edge of a low-level pulse generated by the monostable multivibrator. The outputs of both the IRQ and ITACK Flags are sent through buffers with open collector outputs to provide wired-OR signals as specified in Reference 3.

Global Bus Interface Section

The Global Bus interface section consists of the Bus Length Register, Bus Position Register, Output Data Buffer, and the Bus



*All connections not shown.

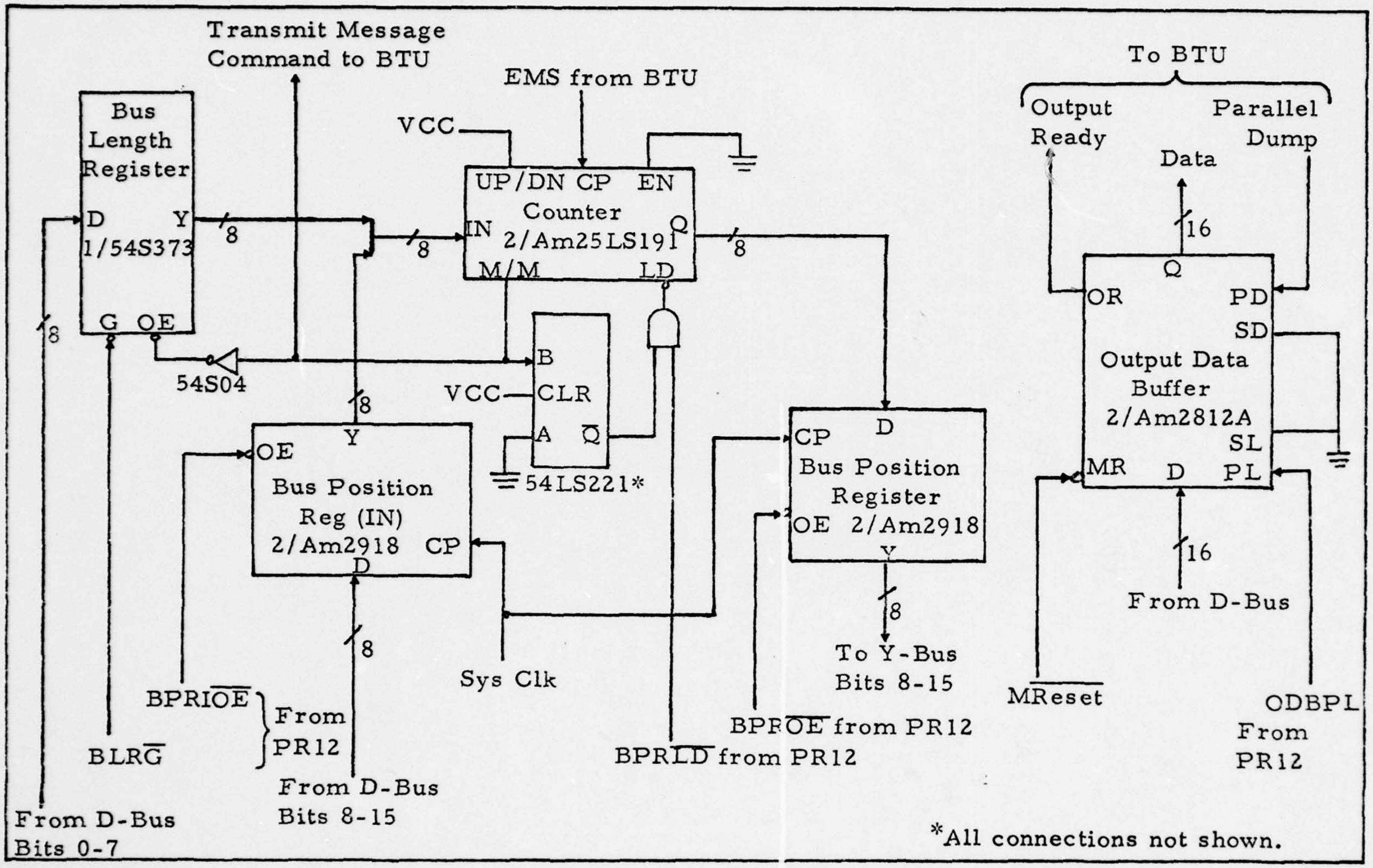
Fig. 29. I-Bus Interface--CPU Interrupt Control Logic

Translation Unit. The Bus Translation Unit was not part of this design and will not be described.

Bus Length Register. The Bus Length Register consists of one 54S373 octal D-type transparent latch with three-state outputs (Figure 30). Inputs to this register is from the D-Bus, bits zero through seven. The output of the register is used to load the Bus Position Register down counter, and is enabled when the counter decrements to zero (Min/Max output goes high).

Bus Position Register. The circuit referred to as the Bus Position Register actually consists of three units, the Bus Position Register (In), a down counter, and the Bus Position Register (Figure 30). The Bus Position Register (In) is an 8-bit register consisting of two Am2918 quad D registers. Inputs to this register come from the D-Bus, bits eight through 15. The three-state output, which is used to initially load the counter with the preset bus position, is enabled when a $\overline{\text{BPRIOE}}$ is issued by the microinstruction.

The down counter consists of two Am25LS191 synchronous 4-bit binary up-down counters. It is initially loaded with the contents of the Bus Position Register (In) when a $\overline{\text{BPRLD}}$ signal is issued by the microinstruction. The counter is decremented each time an EMS signal is received from the BTU. When the counter decrements to zero, a high signal is issued from the Min/Max output which is a Transmit Message command to the BTU. The Min/Max output also causes the contents of the Bus Length Register to be loaded into the counter. The Min/Max signal goes to the Output Enable input of the Bus Length Register through an inverter, 54S04. The signal also triggers a monostable multivibrator, 54LS221, which generates a low-level pulse to the Load input of the counter.



*All connections not shown.

Fig. 30. Global Bus Inteface Section (Excluding Bus Translation Unit)

This pulse is delayed approximately 35 nanoseconds which allows enough time for the contents of the Bus Length Register to reach the counter input. When the Min/Max signal is sent to the Bus Length Register, there is approximately a five nanosecond delay through the inverter and approximately an 18 nanosecond delay for the contents of the register to be released.

The Bus Position Register consists of two Am2918's. Its input is the output of the counter. Its three-state output goes to bits eight through 15 of the Y-Bus when a $\overline{\text{BPROE}}$ signal is issued by the microinstruction.

Output Data Buffer. The Output Data Buffer is used to transfer outgoing messages to the Output Shift Register in the BTU. The buffer consists of two Am2812A 32-word by 8-bit first-in first-out (FIFO) memories (Figure 30). The FIFO memory is used because of the asynchronous operation between the BIU and the Global Bus and of the BIU's requirement to simultaneously transmit and receive on one global bus and monitor the alternate bus. With the use of a FIFO memory (buffer), the BIU can place the outgoing message in the memory when the BIU is ready and the BTU can output the message when it is ready. The Am2812A allows the BIU to write new data into the buffer at the same time that the BTU is reading from the buffer without requiring any synchronization between the two.

A 16-bit parallel word is written into the buffer from the D-Bus when the microinstruction issues an ODBPL signal. The word is read from the buffer when the BTU issues a PD signal. (If desired, the buffer can be configured for serial output.)

Summary

This chapter presented a general description of the BIU hardware design. The next step in the BIU design was to define the microword and to develop the RTL description of the BIU microroutines. These are presented in the next chapter. Although these design steps are presented sequentially in this report, there were many iterations between the hardware and software developments.

IV MICROCODING

Introduction

After the BIU hardware was defined, the microwrod required for the BIU operation was developed. This chapter describes the microword format and introduces the microroutines and microcodes presented in Appendices B and C.

Microword Format

The microword is a 92-bit word consisting of 12 fields (Figure 31). The first field, Next Address Select Instruction Field, is used to provide a four-bit instruction to the Am2910 Microprogram Controller (Table V). The instruction selects the source for the Am2910 "Y" outputs (Microprogram PROM address) and controls the three enable signals $\overline{PL}$, $\overline{MAP}$, and $\overline{VECT}$. For each instruction, one and only one of these enable signals is low. These signals are used to enable the outputs of the Next Address Field of the Pipeline Register, the CAW Mapping PROM Register, and the Vector Mapping PROM, respectively.

The second field, Condition Code Multiplexer Select Field, is a 5-bit field used to select one of 27 inputs to the Condition Code Multiplexer. Table VI lists the inputs to the multiplexer.

The third field, Next Address Field, is a 12-bit field used to provide an address for a microprogram jump.

The next three fields are used to provide the ALU (Am2901) instruction. The 3-bit ALU Source Field is used to select one of the eight source operand pairs available to the ALU (Table VII). The 3-bit ALU Function Field is used to select the function to be performed, five

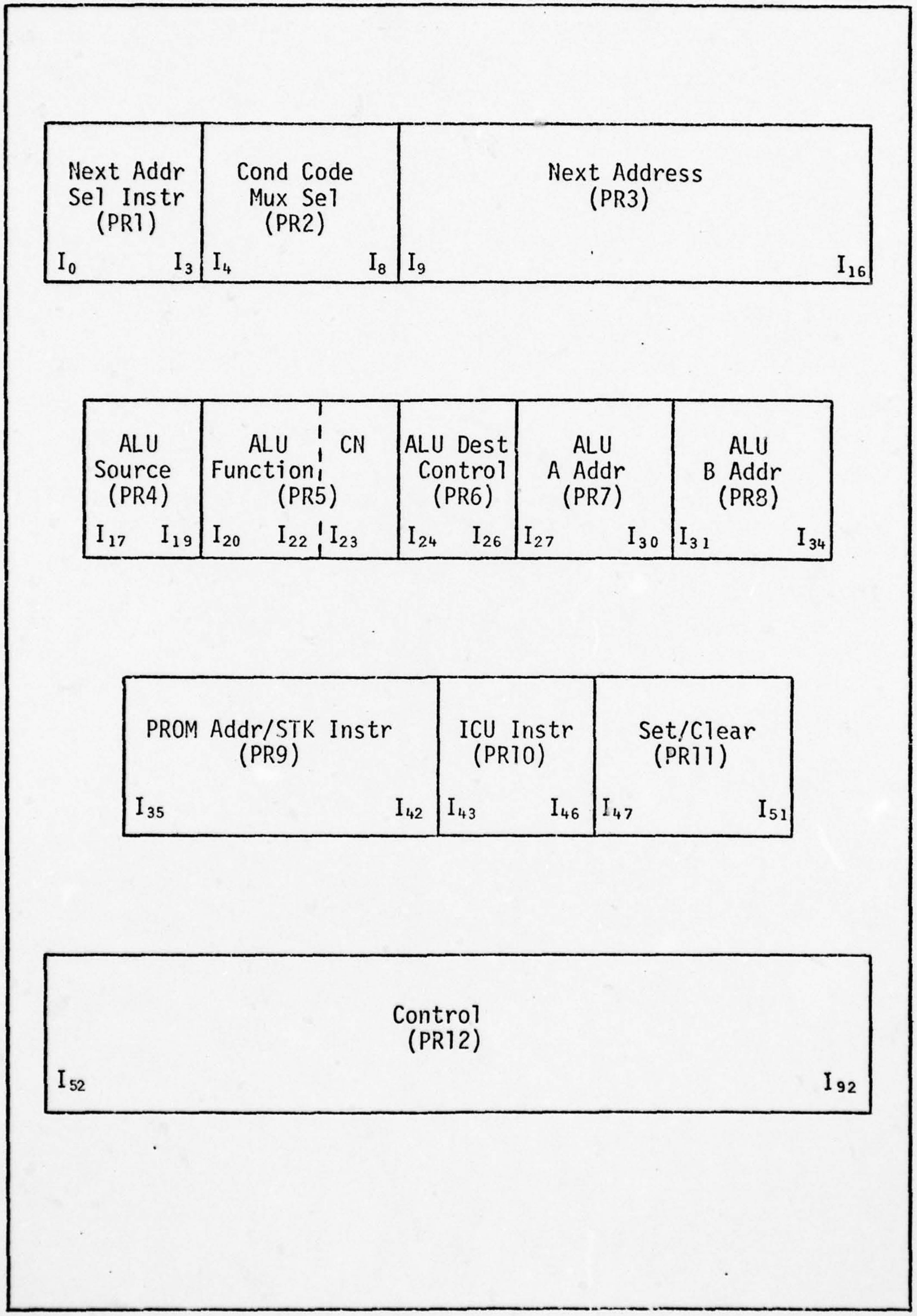


Figure 31. Microword Format

TABLE V

Next Address Select Instruction Field

Microcode (Decimal) $I_0 - I_3$	Mnemonic	Name	Reg/Cntr Contents	$\overline{CCEN}$ = LOW and $\overline{CC}$ = HIGH		$\overline{CCEN}$ = HIGH or $\overline{CC}$ = LOW		Reg/ Cntr	Enable
				Y	Stack	Y	Stack		
0	JZ	Jump Zero	X	0	Clear	0	Clear	Hold	PL
1	CJS	Cond JSB PL	X	PC	Hold	D	Push	Hold	PL
2	JMAP	Jump Map	X	D	Hold	D	Hold	Hold	MAP
3	CJP	Cond Jump PL	X	PC	Hold	D	Hold	Hold	PL
4	PUSH	Push/Cond LD Cntr	X	PC	Push	PC	Push	Note 1	PL
5	JSRP	Cond JSB R/PL	X	R	Push	D	Push	Hold	PL
6	CJV	Cond Jump Vector	X	PC	Hold	D	Hold	Hold	VECT
7	JRP	Cond Jump R/PL	X	R	Hold	D	Hold	Hold	PL
8	RFCT	Repeat Loop, Cntr ≠ 0	≠0	F	Hold	F	Hold	Dec	PL
			=0	PC	Pop	PC	Pop	Hold	PL
9	RPCT	Repeat PL, Cntr ≠ 0	≠0	D	Hold	D	Hold	Dec	PL
			=0	PC	Hold	PC	Hold	Hold	PL
10	CRTN	Cond RTN	X	PC	Hold	F	Pop	Hold	PL
11	CJPP	Cond Jump PL & Pop	X	PC	Hold	D	Pop	Hold	PL
12	LDCT	LD Cntr & Continue	X	PC	Hold	PC	Hold	LOAD	PL
13	LOOP	Test End Loop	X	F	Hold	PC	Pop	Hold	PL
14	CONT	Continue	X	PC	Hold	PC	Hold	Hold	PL
15	TWB	Three-way Branch	≠0	F	Hold	PC	Pop	Dec	PL
			=0	D	Pop	PC	Pop	Hold	PL

NOTE 1:. If $\overline{CCEN}$ = LOW and $\overline{CC}$ = HIGH, Hold; else load. X = Don't Care

TABLE VI

Condition Code Multiplexer Select Field

Micro Code (Decimal) $I_4 - I_8$	Mnemonic	Description
0	TRUE	True Signal
1	INTR	Interrupt request
2	NEG	ALU output negative
3	ZERO	ALU output zero
4	$\overline{\text{ZERO}}$	ALU output not zero
5	$\overline{\text{BGI}}$	Bus Grant (In) signal low
6	TACK	Transfer Acknowledge signal high
7	$\overline{\text{DTACK}}$	Delayed Transfer Acknowledge signal low
8	IAK	Interrupt Acknowledge signal high
9	INHB	Inhibit signal high
10	CIVBOR	CPU Interrupt Vector Buffer output ready
11	MHS	Message Header Sync received
12	EMS(R)	End Message Sync received
13	EMS(T)	End Message Sync transmitted
14	GMHE	Global message header error
15	OP	Output pending
16	RB	Response bit set
17	GHP	Global high priority message
18	GBDD	Global bus dominance disabled
19	OE	Output enabled
20	GHPMRM	GHPMR interrupt masked
21	GMLVM	GMLV interrupt masked
22	GMHEM	GMHE interrupt masked
23	GMWCEM	GMWCE interrupt masked
24	GTTOM	GTTO interrupt masked
25	GMRM	GMR interrupt masked
26	GTCM	GTC interrupt masked

AD-A053 269 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 9/2
BUS INTERFACE UNIT DESIGN FOR THE DISTRIBUTED PROCESSOR/MEMORY --ETC(U)
DEC 77 L T KONNO
UNCLASSIFIED AFIT/GE/EE/77-24 NL

2 OF 3
AD
A053269

logic and three arithmetic (Table VIII). Added to this field is another bit (I_{23}) for the ALU carry input, CN. This input affects the ALU results when in the arithmetic mode as shown in Table IX. The 3-bit Destination Control Field selects the ALU destination register (Table X).

The seventh and eighth fields are used to provide the addresses for the A and B ports of the ALU RAM. The 4-bit ALU A Address Field is used to select one register whose contents are brought to the A-port. The 4-bit ALU B Address Field is used to select one register whose contents are brought to the B-port. The ALU B Address Field is also used to select the register into which new data is to be written. The registers in the ALU RAM were listed in Table IV (Chapter III).

The ninth field, PROM Address/Stack Instruction Field, is a shared field used to provide either the address for the BIU PROM or the instruction for the BIU Stack (Am2930). The PROM addresses and the stack instructions are given in Table XI.

The tenth field is a 4-bit field used to provide the instruction for the Am2914 Interrupt Control Unit. The instructions for the Am2914 are given in Table XII.

The last two fields are used for control. The 5-bit Set/Clear Field encodes 32 control signals used to set and clear various flags in the BIU (Table XIII). The 40-bit Control Field is used to provide the control signals listed in Table XIV. As stated in Chapter III, to provide system flexibility and maximum operational speed (features of prime importance during the development of this system), no attempt was made to shorten this field by encoding or overlapping with other microinstruction fields. Optimization of the microword should be accomplished after the BIU is operating and all timing requirements satisfied.

TABLE VII

ALU Source Operand Field

Microcode (Decimal) I_{17}-I_{19}	ALU Source Operands R	 S
0	A	Q
1	A	B
2	0	Q
3	0	B
4	0	A
5	D	A
6	D	Q
7	D	0

Notes: "A" designates RAM A-port
"B" designates RAM B-port
"Q" designates Q Register
"D" designates direct data input

TABLE VIII
ALU Function Field

Microcode (Decimal) I_{20}-I_{22}	ALU Function	Symbol
0	R Plus S	$R + S$
1	S Minus R	$S - R$
2	R Minus S	$R - S$
3	R OR S	$R \vee S$
4	R AND S	$R \wedge S$
5	$\overline{R}$ AND S	$\overline{R} \wedge S$
6	R EX-OR S	$R \veebar S$
7	R EX-NOR S	$\overline{R \veebar S}$

TABLE IX

ALU Arithmetic Mode Functions

Microcode (Decimal)		CN = 0(Low)		CN = 1(High)	
I_{17-19},	I_{20-22}	Group	Function	Group	Function
0	0	ADD	A+Q	Add plus one	A+Q+1
1	0		A+B		A+B+1
5	0		D+A		D+A+1
6	0		D+Q		D+Q+1
2	0	PASS	Q	Increment	Q+1
3	0		B		B+1
4	0		A		A+1
7	0		D		D+1
2	1	Decrement	Q-1	PASS	Q
3	1		B-1		B
4	1		A-1		A
7	2		D-1		D
2	2	1's Comp.	-Q-1	2's Comp. (Negate)	-Q
3	2		-B-1		-B
4	2		-A-1		-A
7	1		-D-1		-D
0	1	Substract (1's Comp)	Q-A-1	Substract (2's Comp)	Q-A
1	1		B-A-1		B-A
5	1		A-D-1		A-D
6	1		Q-D-1		Q-D
0	2		A-Q-1		A-Q
1	2		A-B-1		A-B
5	2		D-A-1		D-A
6	2		D-Q-1		D-Q

TABLE X

ALU Destination Control Field

Microcode (Decimal)	RAM Function		Q-Reg. Function		Output
I_{24} - I_{26}	Shift	Load	Shift	Load	Y
0	X	None	None	F-Q	F
1	X	None	X	None	F
2	None	F-B	X	None	A
3	None	F-B	X	None	F
4	Down	F/2-B	Down	Q/2-Q	F
5	Down	F/2-B	X	None	F
6	Up	2F-B	Up	2Q-Q	F
7	Up	2F-B	X	None	F

Notes: "X" designates Don't care
"F" designates internal ALU output
"Up" is towards most significant bit
"Down" is towards least significant bit

TABLE XI

PROM Addr/Stack Instruction Field

PROM Addr Mnemonic	Value (Hexidecimal)	Function
OCon	0000	Constant, 0_{10}
8Con	0008	Constant, 8_{10}
WIBF	0007	Word used for Direct Memory Access Write operation. Three least significant bits contain complement of $\overline{TRQ}$, $\overline{IOSL}$, & $\overline{DRCV}$.
RIBF	0006	Word used for Direct Memory Read Operation. Three least significant bits contain complement of $\overline{TRQ}$, $\overline{IOSL}$, & DRCV.
$\overline{BDM}$	D777	Constant used to clear GBD mask
MIDZC	003F	Constant used to clear ten most significant bits
MIDC	To be deter.	Constant used to obtain MIAMM address. Seventh and eighth least significant bits contain value, other bits are zero.
MBZC	03FF	Constant used to clear six most significant bits
MBC	To be deter.	Constant used to obtain Message Buffer address. Six most significant bits contain value, other bits are zero.
BDM	2000	Constant used to set GBD mask
GBQIV	To be deter	GBO Interrupt vector
GMWLEIV	↓	GMWLE interrupt vector
GMPEIV	↓	GMPE interrupt vector
GMEEIV	↓	GMEE interrupt vector
GTTOIV	↓	GTTO interrupt vector
GHPMRIV	↓	GHPMR interrupt vector
GMLVIV	↓	GMLV interrupt vector

TABLE XI
(Continued)

PROM Addr Mnemonic	Value (Hexidecimal)	Function
GMHEIV	To be deter.	GMHE interrupt vector
GMWCEIV	↓	GMWCE interrupt vector
GMRIV	↓	GMR interrupt vector
GTCIV	↓	GTC interrupt vector
IM	7FF0	Constant used to process interrupt mask
OAFW	0002	Constant used to set Output Active Flag
$\overline{\text{OAFW}}$	FFFD	Constant used to clear Output Active Flag
OPFW	0004	Constant used to set Output Pending Flag
$\overline{\text{OPFW}}$	FFFB	Constant used to clear Output Pending Flag
OECon	0001	Constant used to set Output Enable Flag
$\overline{\text{OECon}}$	FFFE	Constant used to clear Output Enable Flag
MIAMM00	To be deter.	MIAMM addresses
MIAMM63	↓	↓
STK Instr		
PD	000C	Push D, push "D" input onto stack
PS	000D	Pop S, pop stack

TABLE XII

Interrupt Control Unit Instruction Field

Microcode (Decimal) $I_{43}-I_{46}$	IE	Function
0	0	Master Clear
1	0	Clear All Interrupts
2	0	Clear Interrupts via Y-Bus
3	0	Clear Interrupt via Mask Register
4	0	Clear Interrupt, Last Vector Read
5	0	Read Vector
6	0	Read Status Register
7	0	Read Mask Register
8	0	Set Mask Register
9	0	Load Status Register
10	0	Bit Clear Mask Register
11	0	Bit Set Mask Register
12	0	Clear Mask Register
13	0	Disable Request
14	0	Load Mask Register
15	0	Enable Request
X	1	Instruction Disable

Note: "X" designates Don't Care

TABLE XIII

Set/Clear Field Signals

Mnemonic	Name
$\overline{\text{WRCL}}$	Word Received Interrupt Clear
$\overline{\text{GMWLECL}}$	Global Message Word Length Error Intr Clr
$\overline{\text{GMPECL}}$	Global Message Parity Error Intr Clear
$\overline{\text{GMEECL}}$	Global Message Encoding Error Intr Clear
$\overline{\text{STAB}}$	Switch to Alternate Bus
$\overline{\text{MHSCL}}$	Message Header Sync Clear
$\overline{\text{EMS(R)CL}}$	End Message Sync (Received) Clear
$\overline{\text{EMS(T)CL}}$	End Message Sync (Transmitted) Clear
GMHE	Global Message Header Error
OP	Output Pending
$\overline{\text{OPCL}}$	Output Pending Clear
RB	Response Bit
$\overline{\text{RBCL}}$	Response Bit Clear
GHP	Global High Priority
$\overline{\text{GHPCL}}$	Global High Priority Clear
GBDD	Global Bus Dominance Disabled
$\overline{\text{GBDDCL}}$	Global Bus Dominance Disabled Clear
OE	Output Enable
$\overline{\text{OECL}}$	Output Enable Clear
$\overline{\text{GHPMRI}}$	Global High Priority Message Received Intr
$\overline{\text{GMLVI}}$	Global Message Length Violation Interrupt
$\overline{\text{GMHEI}}$	Global Message Header Error Interrupt

TABLE XIII

(Continued)

Mneomonic	Name
$\overline{\mathrm{GMWCEI}}$	Global Message Word Count Error Interrupt
$\overline{\mathrm{GTCI}}$	Global Transmission Completed Interrupt
GTCI	Global Transmission Completed Interrupt
$\overline{\mathrm{GTCCL}}$	Global Transmission Completed Clear
BRQ	Bus Request
$\overline{\mathrm{TACK}}$	Transfer Acknowledge
$\overline{\mathrm{IRQ}}$	Interrupt Request
$\overline{\mathrm{IRQCL}}$	Interrupt Request Clear
$\overline{\mathrm{ITACK}}$	Interrupt Transfer Acknowledge
$\overline{\mathrm{ITACKCL}}$	Interrupt Transfer Acknowledge Clear

TABLE XIV

Control Field Signals

Mnemonic	Name
$\overline{\text{BTUOE}}$	Bus Translation Unit Output Enable
$\overline{\text{BLRG}}$	Bus Length Register Enable
$\overline{\text{BPRIOE}}$	Bus Position Register (In) Output Enable
$\overline{\text{BPRLD}}$	Bus Position Register Load
$\overline{\text{BPROE}}$	Bus Position Register Output Enable
ODBPL	Output Data Buffer Parallel Load
$\overline{\text{SCDG}}$	Set/Clear Decoder Enable
$\overline{\text{ICUIE}}$	Interrupt Control Unit Instruction Enable
$\overline{\text{CCLG}}$	Condition Code Latch Enable
$\overline{\text{MCRLD}}$	Microprogram Controller Register Load
MCCI	Microprogram Controller Carry-In
$\overline{\text{PR9OE}}$	Pipeline Register 9 Output Enable
$\overline{\text{PROMCS}}$	BIU PROM Select
PARLE	PROM Address Register Later Enable
$\overline{\text{PAROE}}$	PROM Address Register Output Enable
$\overline{\text{ALUOE}}$	ALU Output Enable
$\overline{\text{RBRG}}$	Response Bit Register Enable
$\overline{\text{RBRDOE}}$	Response Bit Register Decoder Output Enable
$\overline{\text{STKIEN}}$	BIU Stack Instruction Enable
$\overline{\text{STKOE}}$	BIU Stack Output Enable
CIVBPL	CPU Interrupt Vector Buffer Parallel Load

TABLE XIV
(Continued)

Mnemonic	Name
CIVBPD	CPU Interrupt Vector Parallel Dump
$\overline{\text{IQPRCP}}$	Input Queue Pointer Register Clock Pulse
$\overline{\text{IQPROE}}$	Input Queue Pointer Register Output Enable
$\overline{\text{IBFRLE}}$	I-Bus Flag Register Latch Enable
$\overline{\text{IBFRBE}}$	I-Bus Flag Register Bus Enable
$\overline{\text{IBFRCL}}$	I-Bus Flag Register Flag Clear
$\overline{\text{IBARLE}}$	I-Bus Address Register Latch Enable
$\overline{\text{IBARBE}}$	I-Bus Address Register Bus Enable
$\overline{\text{IBARCL}}$	I-Bus Address Register Flag Clear
$\overline{\text{IBARRLE}}$	I-Bus Address Register Rcvr Latch Enable
$\overline{\text{IBAROE}}$	I-Bus Address Register Output Enable
$\overline{\text{IBDRLE}}$	I-Bus Data Register Latch Enable
$\overline{\text{IBDRBE}}$	I-Bus Data Register Bus Enable
$\overline{\text{IBDRCL}}$	I-Bus Data Register Flag Clear
$\overline{\text{IBDRRLE}}$	I-Bus Data Register Receiver Latch Enable
$\overline{\text{IBDROE}}$	I-Bus Data Register Output Enable

NOTE: Three bits of the field are not used.

Microroutines

The microroutines and associated microcodes required for the BIU operation are presented in Appendices B and C respectively. Table B-1 lists the microroutines under the six primary BIU functions that they support. Table B-2 lists the microroutines under four categories:

(1) service routines, (2) external interrupt routines, (3) program generated interrupt routines, and (4) CAW routines.

The service routines are used for system initialization and for supporting the other routines. During power-on, the PE CPU issues a Master Reset (MReset) signal which clears the various BIU Flags and registers including the Pipeline Register 1 (PR1) which contains the instruction for the Microprogram Controller. When PR1 is cleared, it presents the Zero instruction to the Microprogram Controller which causes a jump to the Power-Up routine, a one-word routine located at address Zero in the Microprogram PROM. This routine performs a master clear in the Interrupt Control Unit, clears the CPU Interrupt Status Register in the ALU RAM, and performs a jump to the Wait for Interrupt routine.

As stated in Chapter III, the BIU is an interrupt driven microprocessor. The Wait for Interrupt routine keeps the BIU in a loop waiting for an external interrupt. When an external interrupt occurs, a jump is made to the Service Interrupt routine which saves certain information (Figure B-3) and performs a jump to the appropriate external interrupt routine. The external interrupt routine may call on a service routine or may perform a jump to a program generated interrupt routine or a CAW routine (Table B-2). At the end of these routines, a jump is made to the Return routine which returns the BIU to the state at which it was at the time of the interrupt. Prior to the actual return, a check is made for the occurrence of any higher priority interrupts. These interrupts are serviced before returning to the original state.

The structure of the program generated interrupt routines should be noted. Five of the routines service interrupts of the enable/disable class (see Chapter II, Interrupt Interface Control Function) and have two entry points. As an example, refer to the Global High Priority Message Received (GHPMR) routine (Figure B-16). When the message reception microprogram detects that a high priority message has been received, it calls on the GHPMR routine (jumps to GHPMR00) to process a GHPMR interrupt to the PE CPU, if the interrupt is not masked. If the interrupt is masked, the GHPMR routine presents a GHPMR interrupt to the Interrupt Control Unit. When the PE CPU clears the mask, the GHPMR interrupt is processed through the Interrupt Control Unit and the GHPMR routine is entered at the second entry point GHPMRI (a jump to GHPMRI00). The PE CPU is then interrupted.

Summary

This chapter described the microword format and the microroutines and microcodes developed for the BIU operation. Microcoding was the last step in the design. The next chapter presents the discussion and conclusions of this design effort.

V. DISCUSSION AND CONCLUSIONS

Discussion

Design Summary. The BIU is designed as an interrupt driven microprocessor. As such, it does not operate from a set of macroinstructions, but is driven by interrupt stimuli which cause the BIU to execute the appropriate interrupt microroutine. To improve the BIU performance in the execution of these routines, first level pipelining is employed. This is a technique in which the microinstruction for the next cycle is fetched from microprogram memory on an overlapping basis with the execution of the current microinstruction.

The BIU is divided into four major sections. First is the Global Bus Interface Section which interfaces the BIU with the two Global buses (in the Local BIU, the corresponding unit interfaces the BIU with only one Local Bus). The second section is the Control Section. Two major units of this section are the Interrupt Control Unit and the Microprogram Controller. The Interrupt Control Unit receives the interrupts, masks the appropriate interrupts, determines the highest priority interrupt, produces an appropriate interrupt vector, and notifies the Microprogram Controller of the interrupt. The Microprogram Controller selects the microinstruction to be executed. It consists of one Am2910 40-pin chip which is an address sequencer with the capability to provide conditional branching to any microinstruction within a 4096-microword range. It has a last-in first-out stack which provides microsubroutine return linkage and looping capability. It also contains a microinstruction loop count control. The third section is the Data Manipulation Section. The major unit of this section is the Arithmetic Logic

Unit which performs the data manipulation required for identifying messages and for determining memory addresses. The registers required for processing messages are located in this unit. The fourth section is the I-Bus Interface Section which interfaces the BIU with the I-Bus.

This design uses TTL logic. A bipolar bit-slice microprocessor family is employed. To improve system efficiency, combinations of standard, high-speed, and low-power devices are used in the various BIU sections. Timing was taken to be the driving factor in device selection. When timing was critical, high-speed devices were selected; otherwise, standard and low-power devices were selected. Approximately 130 integrated circuit chips are used in the design.

The minimum system clock cycle that can be used in the BIU is 201 nanoseconds. This value is determined by the component delays along the longest "pipeline register clock to logic to pipeline register clock" path (Ref 8:Sec 1, 20). The longest signal path is determined to be from the Pipeline Register 2(13 nsec), through the 54S04 inverter (5 nsec) at the Condition Code Multiplexer select input, the Condition Code Multiplexer (35 nsec), the 54S08 AND gate at the multiplexer output (8 nsec), the Microrpogram Controller (60 nsec), and the microprogram PROM to the Pipeline Register 9 (80 nsec), refer to Figures 20, 18, and 15. To be conservative, it is recommended that during implementation, a system clock period of 250 nanoseconds be used.

Specification Deviations. As stated in the previous chapters, the BIU design contains some deviations from the specification. These deviations resulted from changes made by the Air Force Avionics

Laboratory, need for improved performance, and new requirements resulting from the use of a bit-slice microprocessor rather than random logic for the design. This section will summarize these specification deviations.

In this design, the serial-to-parallel data conversion, parallel-to-serial data conversion, and the watchdog timer mechanisms are not included in the message reception control function, message transmission control function, and bus access control function respectively, as called for in the specifications. The Air Force Avionics Laboratory included these tasks in the Bus Translation Unit (bus data translation function) which they are designing.

The Message Identity Associative Match Map (MIAMM) is placed in the BIU PROM rather than the PE memory. This is done to increase BIU operational speed in message reception and to reduce I-Bus utilization, which provides more bus access time to the other PE modules. If the MIAMM was left in the PE memory, whenever a message header is received, either global or local, the appropriate BIU would have to gain access to the I-Bus and then fetch the MIAMM word from PE memory. If the bus is in use, a slight delay would be incurred waiting for the bus to become available. Also, accessing the I-Bus to fetch a MIAMM word for each message header received (the BIU must process all message headers transmitted on the Global/Local bus), may mean poor utilization of the I-Bus since it is conceivable that most of the messages on both the Global and Local Buses are not for the PE.

As a result of placing the MIAMM in the BIU PROM, it was necessary to make a slight change in the way the address for the MIAMM word is obtained. Only an 8-bit address is required for the PROM. Therefore, to obtain the address, a 2-bit, instead of a 10-bit, constant is appended to the six most significant bits of the message header's Message Identity field.

For message transmission, an Output Data Buffer is utilized rather than an Output Data Register. The BIU Specifications specify the use of an Output Data Register to provide single-word buffering for message transmission. More than a single-word buffer is required in this design because a microprocessor is employed and there are requirements for simultaneous operations. The operation between the BIU and the Global bus is asynchronous and the BIU is required to be able to simultaneously transmit and receive on one bus and monitor an alternate bus. A first-in first-out memory is used as the buffer. The buffer allows the BIU to load the outgoing message when the BIU is ready and allows the BTU to output the message when it is ready (when the PE's allocated time slot arrives). Data can be written into the buffer and read from the buffer at the same time without any kind of synchronization between the two (Ref 9:Sec 7, 28).

The Output Data Buffer also provides more flexibility when transmitting messages. With a single-word buffer, the PE CPU must issue the Activate Output CAW just prior to the arrival of the PE's allocated time slot. The Output Data Buffer allows the PE CPU to issue the Activate Output CAW at any time. The complete message can be written into the buffer and stored until the arrival of the PE's allocated time slot.

In this design, during a CPU interrupt operation the BIU provides the CPU with a trap vector address for the specific interrupt instead of a trap vector address for an interrupt level. In the BIU specifications, the BIU is to provide the CPU with an interrupt level trap vector address when interrupting the CPU. The BIU specifications categorize CPU interrupts under primary level and sub-level interrupts, with a trap vector address assigned to each level. To process interrupts to the CPU, the specifications call for (1) a program-controlled interrupt mask logic for each sub-level interrupt, (2) an Interrupt Flag Register, (3) primary level interrupt priority resolution, generation, and I-Bus interface control logic, and (4) primary level interrupt trap vector address generation logic. The Interrupt Flag Register is to record and retain the occurrence of each interrupt stimulus until the register is read by the PE CPU or a Master Reset is issued by the PE CPU. The interrupt operation is specified as follows. The PE CPU sets the primary level interrupt masks in the primary level interrupt generation control logic (item 3 above) with a Primary Level Interrupt Mask CAW (a hexidecimal 00 word). With the Interrupt Status CAW, the PE CPU loads the sub-level interrupt masks in the Interrupt Status Register. When interrupts occur, the BIU identifies the highest priority interrupt and checks the primary level masks and sub-level masks. If the interrupt is not masked, the appropriate interrupt flag is set in the Interrupt Flag Register and an interrupt request is made to the PE CPU. When the PE CPU completes executing the current macroinstruction, it detects the interrupt request and acknowledges by issuing an Interrupt Acknowledge signal. The BIU then provides the PE CPU with the interrupt level trap vector address in PE memory, which the PE CPU uses to fetch the interrupt handling routine. The PE CPU then reads the contents of the Interrupt Flag

Register in the BIU, and polls the various interrupt flags, servicing those interrupts whose flags are set. It should be noted that each time the PE CPU communicates with the BIU or PE memory, it must gain control of the I-Bus, for which it has the lowest bus access priority. If the other PE modules are utilizing the bus, processing of interrupts is delayed.

In the above interrupt scheme, a considerable amount of overhead is involved in processing interrupts to the PE CPU. In the design developed here, most of the overhead has been eliminated to expedite the processing of interrupts. The Primary Level Interrupt Mask CAW, Input High-Priority Interrupts CAW (a hexidecimal OD word), and the Interrupt Flag Register are not utilized. The Interrupt Status CAW is used to set the interrupt sub-level masks in the BIU. When interrupts occur, the highest priority interrupt is identified and if the interrupt is not masked, an interrupt request is made to the PE CPU. When the interrupt request is acknowledged, the BIU provides the PE CPU with the interrupt trap vector for the specific interrupt. The PE CPU can then access the specific interrupt routine in the PE memory. (The interrupt trap vector address is the same address that the primary level interrupt handling routine would have provided to the PE CPU after identifying the interrupt in the original interrupt handling scheme.)

As seen in Table III, a Transfer Time Out interrupt was added to the CPU interrupts. The Transfer Time Out error is mentioned in the I-Bus Specifications (Ref 3:B-5) but is not listed as an interrupt in the BIU specifications (Ref 3:C-36). It is felt that the PE CPU should be notified when a Transfer Time Out condition occurs.

Due to the addition of the Transfer Time Out interrupt, the contents of the CPU Interrupt Status Register had to be changed to include the mask bit for the Transfer Time Out interrupt. As seen in Figure 12, the mask bit is located between the Message Encoding Error and the Message Received mask bits. Another change made to the CPU Interrupt Status Register (Figure 12) is the grouping together of the interrupt mask bits and the grouping together of the flags associated with message transmission. In the BIU Specifications, the Output Pending Flag bit, which is set by the BIU, is grouped with the interrupt mask bits (Ref 3:C-46).

Additional Work Required. Prior to actual implementation of this BIU design, some additional tasks must be accomplished. First, the interface with the BTU must be finalized. At the time of this design, adequate documentation on the BTU design was not available. The interface signals must be verified and modified as required.

Due to the time constraint on this design effort, detailed timing and loading analyses could not be accomplished. These analyses should be performed to verify that all timing and loading requirements are satisfied and to determine the power requirements of the system. With regard to circuit loading, one of the items that must be considered is the pull-up resistors for all the open collector lines. For the I-Bus open collector lines, all the devices tied to these lines must be identified before the values of the resistors can be determined.

An analysis should also be performed on the operational timing regarding the Global/Local Bus operations and the I-Bus operations. The analysis is requied to verify that all tasks can be accomplished within the specified time under various conditions. To perform the analysis, an estimate must be made on the number and frequency of messages to be received and transmitted. Also, the amount of utilization of the I-Bus by the other PE modules must be estimated. One result of the analysis may indicate that the system operates with enough speed to allow the Message Identity Associative Match Map (MIAMM) to be left in the PE memory. If this is the case, and it is still desirable to have the MIAMM in the PE memory, the BIU design is such that the change can be made without requiring any hardware redesign. All that is required is to change the value of the constant stored in the BIU PROM which is used to obtain the MIAMM address and to rewrite the Message Reception routine.

During system implementation, some other factors that must be considered are noise problems, grounding problems, and termination of unused inputs. This design effort did not reach this level of detail.

This design did not include an interface with a maintenance panel. This feature may be desired to aid in the debugging of the BIU microroutines. It would provide the capability to manually set different interrupt conditions and to manually sequence through the various microroutines. It would also provide the capability to inspect the contents of the various BIU registers.

Recommendations. As a result of this design effort, three recommendations are made to improve system performance. The first recommendation concerns the format of the Command Address Word (CAW), the second concerns a problem associated with the CAW 00, and the third concerns the processing of interrupts to the PE CPU.

The DP/M software should be analyzed to determine whether the CAW's, which are sent over the Address lines of the I-Bus, could be formated so the receiving PE modules can use then as addresses. It should also be determined whether the CAW's for each particular PE module can be assigned consecutive numerical values. These changes would simplify the processing of a CAW, which, in the case of the BIU, would reduce the CAW processing hardware. Presently, the CAW's are command words, and for each PE module or function, the CAW's may not have consecutive numerical values which make them impractical to use as memory addresses. As an example, the BIU Specifications assign hexidecimal values of 00, 03, 05, 07, 09, 0B, and 0D to the CAW's for the Global BIU function (Ref 3:C-37). Also, the CAW's have different meanings depending on the value of the DRCV line (read/write line) of the I-Bus. For CAW's 00, 03, 05, and 07, the DRCV line determines whether a register is to be written into or read from. However, for CAW's 09, 0B, and 0D, the DRCV line gives the CAW a different and unrelated meaning. As an example, CAW 09 with the DRCV line high designates that the CPU is to read the contents of the Bus Position Register. CAW 09 with the DRCV line low designates that the BIU is to switch to the alternate bus.

If the CAW format can be changed, it is recommended that the two or three most significant bits be used to identify the PE module being addressed and that the other bits be used as an address word. (If desired, they could be used as a command word for the other PE modules.) It is not felt that a 16-bit address will be required by any PE module except the PE memory. In this case, as is done now, the IOSL line will be used to designate the PE memory and all 16 address lines will be used for memory address. For the BIU, the two or three most significant bits and the eight least significant bits of the CAW should contain valid data with the other bits filled with zeros. In processing the CAW, the BIU would decode the two or three most significant bits and if the CAW is recognized, the eight least significant bits would be gated to the "D" inputs of the Microprogram Controller and be used as an indirect address for a microroutine to perform the required task.

There is a potential problem in the use of CAW 00 as specified in the BIU Specifications and the I-Bus Specifications. This CAW is used to command more than one PE module. When the PE CPU issues this CAW, the first PE module to decode and identify the CAW will issue a transfer acknowledge signal ($\overline{\text{TACK}}$). Upon receipt of this signal, the PE CPU will not know that the other modules have not yet sent their acknowledgements and it will continue on with the I-Bus transfer operation. This problem has been resolved in this design by not utilizing CAW 00.

During this design, the Am2914 Vectored Priority Interrupt Encoder was found to be a very powerful chip for handling interrupts. Because of the availability of this chip, the following change is proposed to increase the speed of processing interrupts and to reduce the interrupt related hardware and software. The CPU interrupt operation should be revised to allow PE modules to send interrupts directly to the PE CPU without having to mask certain interrupts and without having to serially process the interrupts. If the PE CPU design uses the Am2914 for processing interrupts, the recommended change will involve very little hardware change within the PE CPU. Although the change will increase the number of lines on the I-Bus, it will reduce the hardware in the PE modules and will considerably reduce the supporting software in both the PE executive program and in the BIU microprograms.

In the proposed change, the existing interrupt circuitry on the I-Bus is replaced with interrupt lines from each of the PE modules. This means that there will be 12 lines from each BIU and some other, not yet specified, number of lines from the other PE modules. Since the PE modules should be located together in the operational PE unit, the additional lines on the I-Bus should not create any problems. However, if the number of I-Bus lines must be minimized, the number of interrupt lines can be reduced by encoding the interrupt signals from each PE module.

The following specific advantages are achieved by processing all CPU interrupts through the Am2914's in the PE CPU. First, CPU interrupts are processed faster since the "back-and-forth" transfer of signals between the PE CPU and the PE module is eliminated. Second, the software required to process the interrupts is reduced. All the software associated with loading and reading primary and sub-level interrupt masks are eliminated, most of the PE module software associated with masking and processing the CPU interrupts are eliminated, and all the software associated with reading the Interrupt Flag Register and polling the contents of the register are eliminated. With regard to the BIU, this means the elimination of CAW's 00, 02, 03, 04(R), 0B(R), 0C(R), and the elimination of the microroutines listed in Table XV. These routines make up 52 percent of all the BIU microroutines and account for 29 percent of all the BIU microinstructions. The final advantage is the reduction in hardware in the PE modules. In the BIU, hardware reductions are realized in the Interrupt Control Unit, Condition Code Multiplexer, and Pipeline Register. Also, the BIU Interrupt Vector Buffer and the I-Bus CPU Interrupt Control Logic can be eliminated. These reductions reduce the total number of

TABLE XV

BIU Microroutines to be Eliminated

Service Routines
Interrupt CPU
External Interrupt Routines
Global Bus Quiescent
Global Bus Dominance
Global Message Word Length Error
Global Message Parity Error
Global Message Encoding Error
Global Transfer Time Out
Program Generated Interrupt Routines
Global High Priority Message Received
Global Message Length Violation
Global Message Header Error
Global Message Word Count Error
Global Message Received
Global Transmission Completed
Global Interrupt Acknowledged
CAW Routines
Global Output Interrupt Status
Global Input Interrupt Status

chips by about 10 percent. Because of all these advantages, it is strongly recommended that the CPU interrupt processing operation be revised.

Conclusions

The BIU has been designed as an interrupt driven microprogrammed processor. The one major advantage the design provides over the random logic unit originally proposed in the TI DP/M study is flexibility. Since the BIU will be used in the laboratory to demonstrate a new concept, it is likely that changes will be required in some of its operations. By emulating the BIU functions with a microprocessor, it should be possible to make most changes in software and not require major engineering redesign.

The emulation of the BIU functions was made possible by the use of the Am2900 Bipolar Microprocessor Family. This 4-bit bipolar bit-slice family provided the speed and flexibility required to perform the BIU functions. It allowed the use of a microword length that provided the required instruction execution throughput.

This design provides enough detail to start system implementation. All required microroutines have been developed and coded in mnemonic form. Recommendations have been made to improve system performance.

BIBLIOGRAPHY

1. Air Force Avionics Laboratory. Distributed Processor/Memory Architectures Design Program. AFAL-TR-75-80. Dallas, Texas: Texas Instruments Incorporated, February 1975.

2. Simpson, Robert C. Design of the Bus Interface Unit for the Distributed Processor/Memory System. Master's Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, December 1976.

3. DP/M Preliminary Design. Draft Report. Wright-Patterson Air Force Base, Ohio: Air Force Avionics Laboratory, December 1975.

4. Advanced Micro Devices Incorporated. Am2900 Bipolar Microprocessor Family. Data Book. Sunnyvale, California: Advanced Micro Devices Incorporated, 1976.

5. ----. Am2914 Priority Interrupt Encoder. Pamphlet. Sunnyvale, California: Advanced Micro Devices Incorporated, 1976.

6. ----. Am2910 Microporgram Controller. Pamphlet. Sunnyvale, California: Advanced Micro Devices Incorporated, 1977.

7. ----. Am2930 Program Control Unit. Pamphlet. Sunnyvale, California: Advanced Micro Devices Incorporated, 1977.

8. Mick, John R. and Jim Brick. Microprogramming Handbook. Sunnyvale, California: Advanced Micro Devices Incorporated, 1976.

9. Advanced Micro Devices Incorporated. MOS/LSI Data Book. Sunnyvale, California: Advanced Micro Devices Incorporated, 1976.

Appendix A

Glossary of Abbreviations

ADDR - The 16 address lines of the I-Bus

ALU(CISR) (CPU Interrupt Status Register) - a 16-bit register, located in the BIU Arithmetic Logic Unit RAM, that contains the interrupt masks, which are set by the PE CPU, and the Output Pending flag, Output Active flag, and the Output Enable flag.

ALU(CIVR) (CPU Interrupt Vector Register) - a 16-bit register located in the BIU Arithmetic Logic Unit RAM, that contains the trap vector address in PE memory for the interrupt being processed.

ALU(IAR) (Input Address Register) - a 16-bit register, located in the BIU Arithmetic Logic Unit RAM, that contains the PE memory address during the message reception operation.

ALU(IDR) (Input Data Register) - a 16-bit register, located in the BIU Arithmetic Logic Unit RAM, used to hold incoming message words before they are processed to the PE memory.

ALU(ILR) (Input Length Register) - a 16-bit register, located in the BIU Arithmetic Logic Unit RAM, that contains the length (number of words) of the incoming message.

ALU(OAR) (Output Address Register) - 16-bit register, located in the BIU Arithmetic Logic Unit RAM, that contains the PE memory address during the message reception operation.

ALU(OLR) (Output Length Register) - a 16-bit register, located in the BIU Arithmetic Logic Unit RAM, that contains the length (number of words) of the outgoing message.

BDWT (Bus Dominance Watchdog Timer) - a mechanism in the BTU that monitors the length of each message on the Global/Local bus. It generates an interrupt stimulus if a PE transmits a message of more than eight words.

BGR (Bus Grant) - an I-Bus control signal used to notify a PE module that the module has control of the I-Bus. The Bus Grant line is threaded through each module in a daisy chain configuration. The Bus Grant signal into a module is designated BGRI and the Bus Grant signal out of a module is designated BGRO.

BGRI (Bus Grant In) - See BGR.

BGRO (Bus Grant Out) - See BGR.

BIU (Bus Interface Unit) - the PE module that interfaces the PE with the Global and Local buses in the DP/M System.

BIU PROM - a PROM used to store the CPU interrupt trap vector address words, Message Identity Associative Match Map words, and constants and masks required for processing messages.

BLR (Bus Length Register) - an 8-bit register that contains the number of allocation time slots between the PE's allocated time slot.

BMID (Bus Master Identification) - four I-Bus identification lines used to identify the module in control of the bus.

BPR (Bus Position Register) - an 8-bit register that contains the current position of the PE's allocated time slot in the bus control chain.

BQWT (Bus Quiescence Watchdog Timer) - a mechanism in the BTU that monitors the presence of data transfer activity on the Global/Local bus. It generates an interrupt stimulus if a "no-activity" time of 50 microseconds occurs.

BREL (Bus Release) - an I-Bus control signal which is used to terminate bus master assignment.

BRQ (Bus Request) - an I-Bus control signal which is used to initiate a bus master assignment.

BTU (Bus Translation Unit) - the portion of the BIU which performs the bus data translation function.

CAW (Command Address Word) - a 16-bit command word issued to a PE module by the PE CPU. The word is issued on the address lines of the I-Bus.

CISR (CPU Interrupt Status Register) - a 16-bit register that contains the interrupt masks, whcih are set by the PE CPU, and the Output Pending flag, Output Active flag, and the Output Enable flag. See ALU(CISR).

CIVR (CPU Interrupt Vector Register) - a 16-bit register that contains the trap vector address in PE memory for the interrupt being processed. See ALU(CIVR).

CPU Interrupts - interrupts generated by the BIU that are sent to the PE CPU.

DATA - the 16 data lines of the I-Bus.

DP/M System (Distributed Processor/Memory System) - a system to integrate the avionics onboard an aircraft by using a number of programmable processor/memory elements in a distributed (decentralized) network.

DRCV (Data Receive) - an I-Bus control signal that designates whether the bus master is preparing to send data (DRCV low) or receive data (DRCV high).

EMS (End Message Synchronization) - an invalid Manchester II waveform used to denote the end of a message.

GBD (Global Bus Dominance Interrupt) - a Global interrupt condition indicating that a PE has transmitted a message of more than eight words on the Global Bus. A similar interrupt condition exists for the Local Bus.

GBQ (Global Bus Quiescent Interrupt) - a Global interrupt condition indicating that a "no-activity" time of 50 microseconds has elapsed on the Global Bus. A similar interrupt condition exists for the Local Bus.

GC-field (General Control field) - the second field of the message header. A five-bit field which contains special control/status information used to define or alter the interpretation of the message.

GHPMR (Global High-Priority Message Received Interrupt) - a Global interrupt condition indicating that a high priority message has been received on the Global Bus and requires immediate attention. A similar interrupt condition exists for the Local Bus.

GMEE (Global Message Encoding Error Interrupt) - a Global interrupt condition indicating that a Manchester II encoding error exists in the incoming message word. A similar interrupt condition exists for the Local Bus.

GMHE (Global Message Header Error Interrupt) - a Global interrupt condition indicating that a word length, parity, or encoding error exists in the message header. A similar interrupt condition exists for the Local Bus.

GMLV (Global Message Length Violation Interrupt) - a Global interrupt condition indicating that the BIU is attempting to transmit a message of more than eight words on the Global Bus. A similar interrupt condition exists for the Local Bus.

GMPE (Global Message Parity Error Interrupt) - a Global interrupt condition indicating that a parity error exists in the incoming message word. A similar interrupt condition exists for the Local Bus.

GMR (Global Message Received) - a Global interrupt condition indicating that a Global message has been received. A similar interrupt condition exists for the Local Bus.

GMWCE (Global Message Word Count Error Interrupt)- a Global interrupt condition indicating that an incoming message on the Global Bus does not contain the correct number of data words. A similar interrupt condition exists for the Local Bus.

GMWLE (Global Message Word Length Error Interrupt) - a Global interrupt condition indicating that an incoming data word has an incorrect number of bits. A similar interrupt condition exists for the Local Bus.

GTC (Global Transmission Completed) - a Global interrupt condition indicating that the BIU has transmitted a message on the Global Bus. A similar interrupt condition exists for the Local Bus.

GTTO (Global Transfer Time Out) - a Global interrupt condition indicating that the BIU attempted to address a nonexistent PE memory location or slave. A similar interrupt condition exists for the Local BIU.

H-field (High Priority Message field) - the first field of the message header. A 1-bit field that signifies a high priority message, which requires immediate attention by the PE.

IAK (Interrupt Acknowledge) - an I-Bus control signal issued by the PE CPU acknowledging an interrupt request.

IAR (Input Address Register) - a 16-bit register that contains the PE memory address during the message reception operation. See ALU(IAR).

I-Bus (Internal Bus) - a communication bus, internal to a PE, that connects all the modules of the PE.

IDR (Input Data Register) - a 16-bit register used to hold incoming message words before they are processed to the PE memory. See ALU(IDR).

ILR (Input Length Register) - a 16-bit register that contains the length (number of data words) of the incoming message. See ALU(ILR).

INHB (Interrupt Inhibit) - an I-Bus control signal issued by the PE maintenance panel to inhibit all interrupts to the PE CPU.

IOIU (Input/Output Interface Unit) - the PE module that interfaces the PE with the aircraft sensor.

IOSL (Input/Output Select) - an I-Bus control signal that designates if the module being accessed is PE memory or another PE module. IOSL low designates that PE memory is being accessed.

IQPR (Input Queue Pointer Register) - a 16-bit register that contains the address for the Message Header Input Queue in PE memory.

IRQ (Interrupt Request) - an I-Bus control signal issued by a PE module to notify the PE CPU of an interrupt.

MH (Message Header) - the first word of each message which contains the message routing and control information.

MHS (Message Header Synchronization) - an invalid Manchester II waveform used to denote the start of a message.

MIAMM (Message Identity Associative Match Map) - a 64-word area in the BIU PROM. Each MIAMM word consists of 16-bits and each bit is associated with a unique message header.

MID-field (Message Identity field) - the third field of the message header. A 10-bit field which specifies the identity of the associated message data.

MReset (Master Reset) - an I-Bus control signal used to initialize all PE modules. In the BIU, this signal clears all registers and flags. This signal is called Clear/Reset (CLR) in the BIU Specification.

OA (Output Active) - a flag in the CPU Interrupt Status Register (CISR). When set, it signifies that message transmission is in progress.

OAR (Output Address Register) - a 16-bit register that contains the PE memory address during the message reception operation. See ALU(OAR).

ODB (Output Data Buffer) - a first-in first-out memory used to transfer outgoing messages to the Output Shift Register in the BTU.

OE (Output Enable) - a flag in the CPU Interrupt Status Register (CISR). This flag is controlled by the PE CPU and when set, it signifies that the BIU is allowed to transmit messages.

OLR (Output Length Register) - a 16-bit register that contains the length (number of words) of the outgoing message. See ALU(OLR).

OP (Output Pending) - a flag in the CPU Interrupt Status Register (CISR). When set, it signifies that the BIU is ready to begin message transmission.

PAR (PROM Address Register) - an 8-bit register used when addressing message Identity Associative Match Map words in the BIU PROM.

PE (Processor/Memory Element) - a processing element in the DP/M System which consists of a central processor unit, memory, bus interface unit, and input/output interface unit.

PE CPU (PE Central Processor Unit) - the central processor unit of the PE.

P-field (Parity Field) - the last field of the message header. A 1-bit field that determines odd-parity for the message header.

Sys CLK (System Clock) - a master clock signal that is available on the I-Bus for any device that needs a clock signal. This clock is controlled by the Master Stop signal and is called the Master Clock (MCLK) in the I-Bus Specification.

TACK (Transfer Acknowledge) - an I-Bus control signal sent from a slave to the master indicating that data has been received or sent, depending on whether a Write or Read operation was in progress.

TDM (Time Division Multiplexing) - the transmission of information from several signal channels through one communication system with different channel samples staggered in time to form a composite pulse train.

TRQ (Transfer Request) - an I-Bus control signal that the master sends to a slave indicating a request to read or write.

TTO (Transfer Time Out) - an I-Bus control signal which is issued when a nonexistent memory location or slave is addressed.

WR signal (Word Received signal) - a signal generated by the BTU indicating that a message word has been received.

Appendix B

BIU Microroutines

This appendix presents the functional flowcharts and RTL descriptions of the BIU microroutines. Table B-1 categorizes the microroutines under the six primary BIU functions (see Chapter II) that they support. The table also includes a seventh category, the service routines required for the BIU operation.

Since the BIU was designed as a microprocessor and not as functional hardware modules, the microroutines are reorganized in Table B-2 under the following four types of routines: service routines, external interrupt routines, program generated interrupt routines, and CAW routines. The microroutines that follow are presented in the order listed in Table B-2.

The figures describing the microroutines have two parts, A and B. Part A is a functional flowchart of the routine. Part B is the RTL description of the routine. The codes in the right margin of both parts refer to the Microprogram PROM address of the associated microinstructions. Listings of the microinstructions for the various routines are given in Appendix C.

TABLE B-I

Primary BIU Functions and Supporting Microroutines

Bus Data Translation

Global Bus Quiescent (GBQ, Figure B-9)
Global Bus Dominance (GBD, Figure B-10)

Message Reception Control

Message Reception (MR, Figure B-8)
Global Message Word Length Error (GMWLE, Figure B-11)
Global Message Parity Error (GMPE, Figure B-12)
Global Message Encoding Error (GMEE, Figure B-13)
Global High Priority Message Received (GHPMR, Figure B-16)
Global Message Header Error (GMHE, Figure B-18)
Global Message Word Count Error (GMWCE, Figure B-19)
Global Message Received (GMR, Figure B-20)

Message Transmission Control

Global Message Length Violation (GMLV, Figure B-17)
Global Transmission Completed (GTC, Figure B-21)
Global Activate Output (GAO, Figure B-25)
Global Enable Output (GEO, Figure B-29)
Global Disable Output (GDO, Figure B-30)

Bus Access Control

Global Output Bus Control (GOBC, Figure B-26)

Redundant Bus Management (Global Only)

Switch to Alternate Bus (STAB, Figure B-31)

Processor/Memory Interface

Interrupt CPU (IC, Figure B-5)
Direct Memory Access Write (DMAW, Figure B-6)
Direct Memory Access Read (DMAR, Figure B-7)
Global Transfer Time Out (GTTO, Figure B-14)
Global CAW Interrupt (GCI, Figure B-15)
Global Interrupt Acknowledge (GIA, Figure B-22)
Global Output Interrupt Status (GOIS, Figure B-23)
Global Input Interrupt Status (GIIS, Figure B-24)
Global Input Present Position (GIPP, Figure B-27)
Global Input Queue Pointer (GIOP, Figure B-28)

Service Routines

Power Up (PU, Figure B-1)
Wait For Interrupt (WFI, Figure B-2)
Service Interrupt (SI, Figure B-3)
Return (RTN, Figure B-4)

TABLE B-II

BIU Microroutines

Service Routines

Power Up (PU, Figure B-1)

Wait for Interrupt (WFI, Figure B-2)

Service Interrupt (SI, Figure B-3)

Return (RTN, Figure B-4)

Interrupt CPU (IC, Figure B-5)

Direct Memory Access Write (DMAW, Figure B-6)

Direct Memory Access Read (DMAR, Figure B-7)

External Interrupt Routines

Message Reception (MR, Figure B-8)

Global Bus Quiescent (GBQ, Figure B-9)

Global Bus Dominance (GBD, Figure B-10)

Global Message Word Length Error (GMWLE, Figure B-11)

Global Message Parity Error (GMPE, Figure B-12)

Global Message Encoding Error (GMEE, Figure B-13)

Global Transfer Time OUt (GTTO, Figure B-14)

Global CAW Interrupt (GCI, Figure B-15)

Program Generated Interrupt Routines

Global High Priority Message Received (GHPMR, Figure B-16)

Global Message Length Violation (GMLV, Figure B-17)

Global Message Header Error (GMHR, Figure B-18)

Global Message Word Count Error (GMWCE, Figure B-19)

Global Message Received (GMR, Figure B-20)

Global Transmission Completed (GTC, Figure B-21)

Global Interrupt Acknowledge (GIA, Figure B-22)

CAW Routines

Global Output Interrupt Status (GOIS, Figure B-23)

Global Input Interrupt Status (GIIS, Figure B-24)

Global Activate Output (GAO, Figure B-25)

Global Output Bus Control (GOBC, Figure B-26)

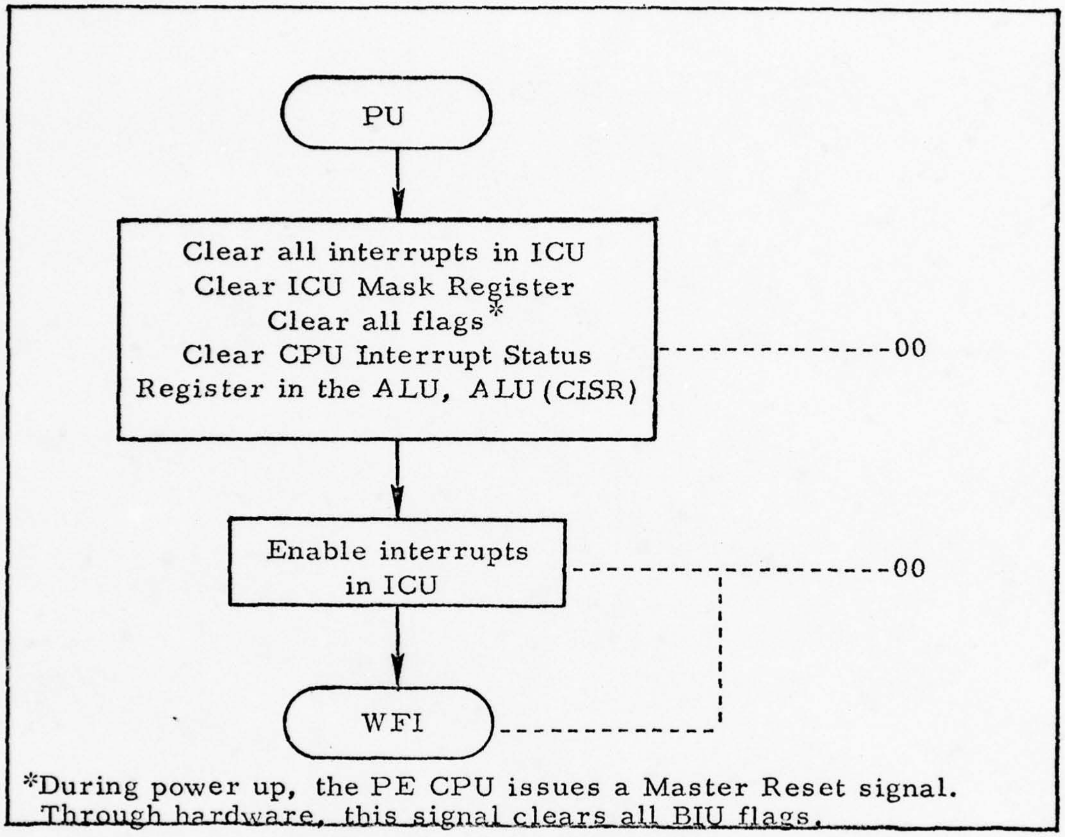
Global Input Present Position (GIPP, Figure B-27)

Global Input Queue Pointer (GIQP, Figure B-28)

Global Enable Output (GEO, Figure B-29)

Global Disable Output (GDO, Figure B-30)

Switch to Alternate Bus (STAB, Figure B-31)


PU
Clear all interrupts in ICU
Clear ICU Mask Register
Clear all flags*
Clear CPU Interrupt Status
Register in the ALU, ALU (CISR)
00
Enable interrupts
in ICU
00
WFI

*During power up, the PE CPU issues a Master Reset signal. Through hardware, this signal clears all BIU flags.

Fig. B-1.A. Functional Flowchart for Power Up Routine (PU)

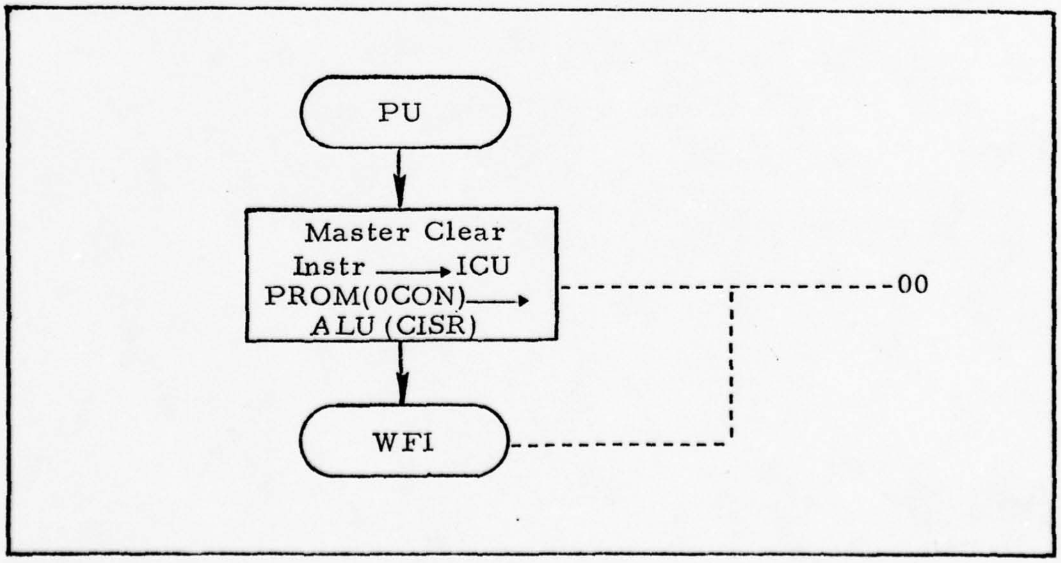


Fig. B-1.B. RTL for Power Up Routine (PU)

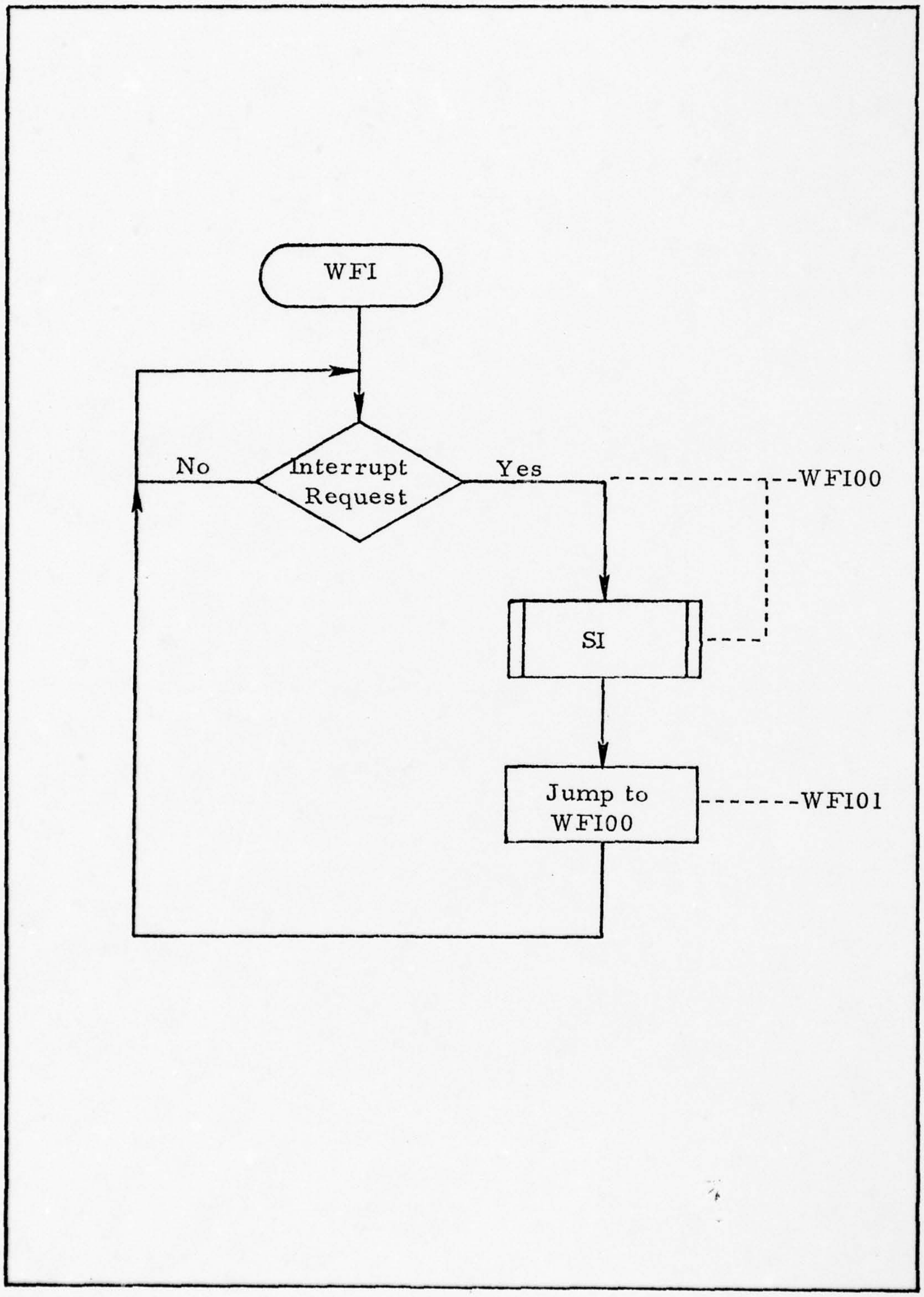


Fig. B-2. Functional Flowchart for Wait for Interrupt Routine (WFI)

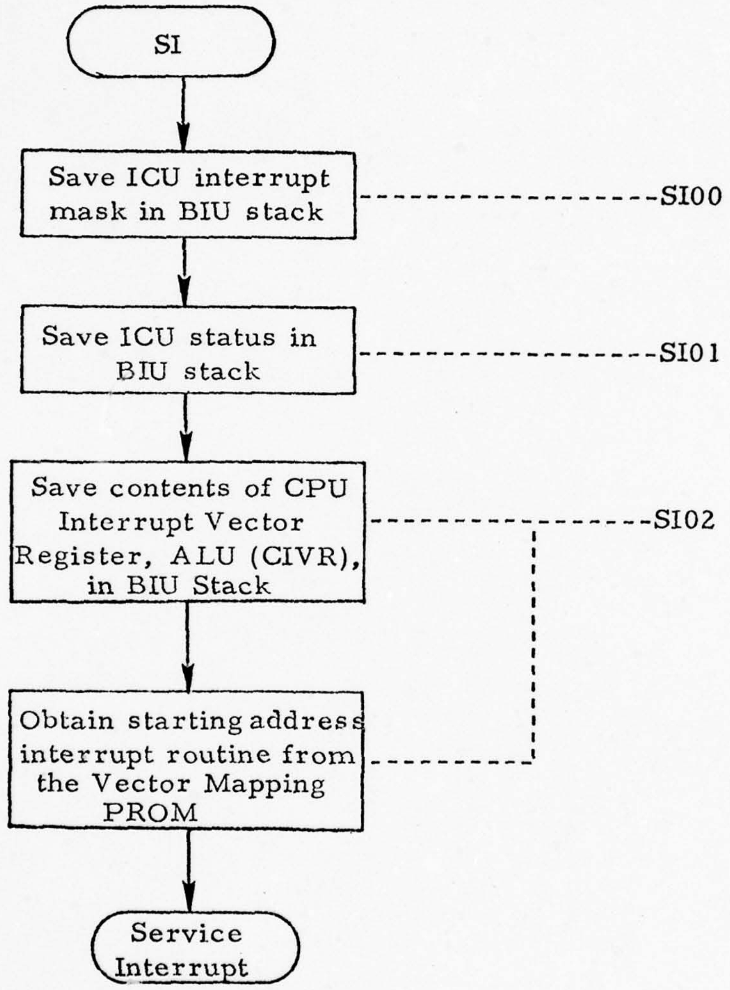

SI
Save ICU interrupt mask in BIU stack
SI00
Save ICU status in BIU stack
SI01
Save contents of CPU Interrupt Vector Register, ALU (CIVR), in BIU Stack
SI02
Obtain starting address interrupt routine from the Vector Mapping PROM
Service Interrupt

Fig. B-3.A. Functional Flowchart for Service Interrupt Routine (SI)

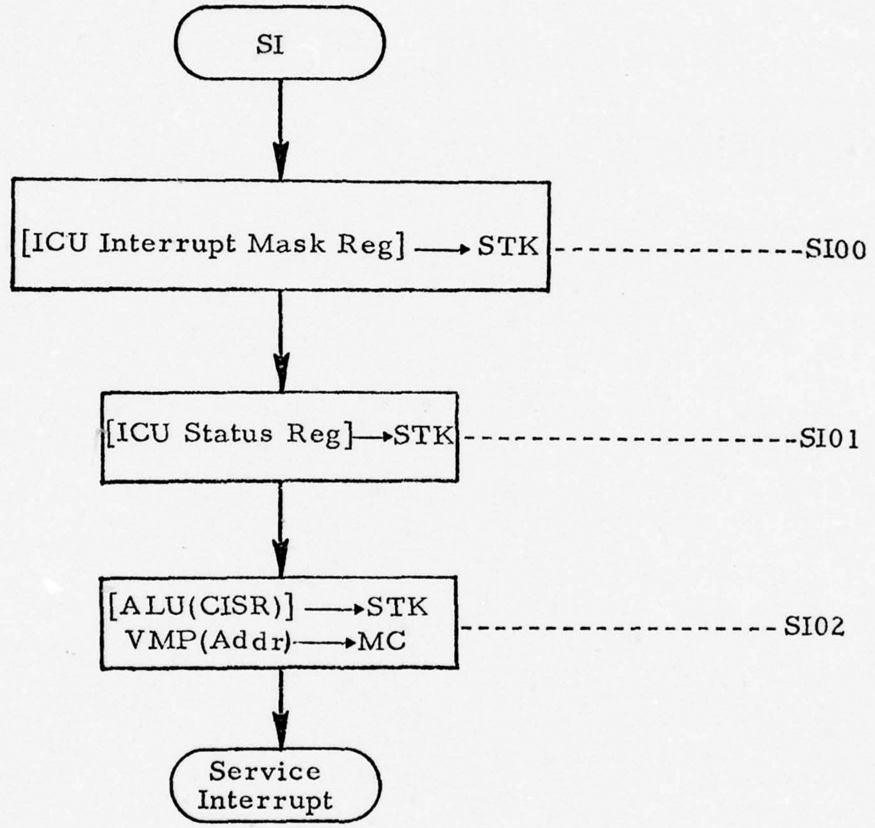


Fig. B-3.B. RTL for Service Interrupt Routine

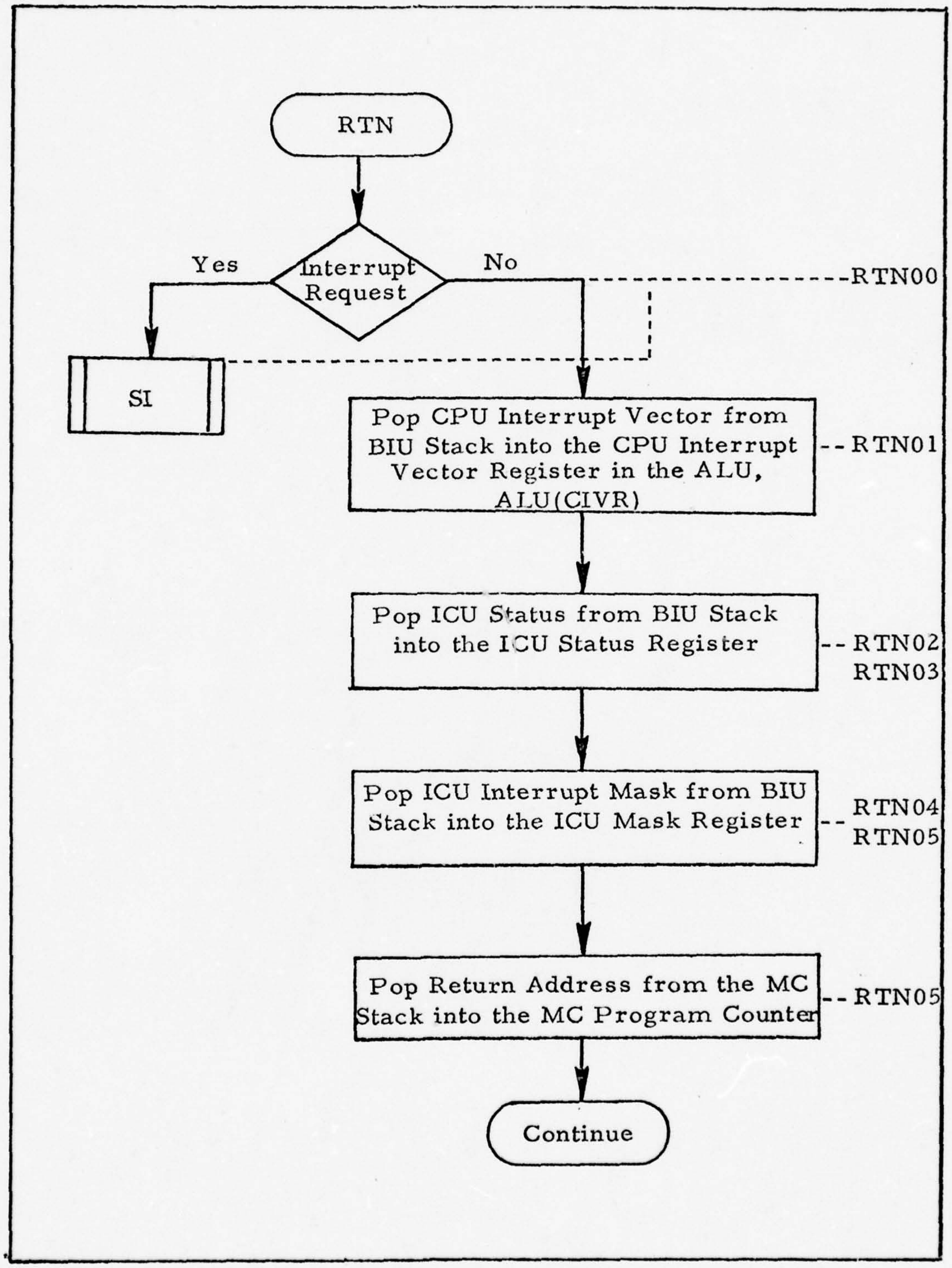

RTN
Interrupt Request
Yes
No
SI
RTN00
Pop CPU Interrupt Vector from BIU Stack into the CPU Interrupt Vector Register in the ALU, ALU(CIVR)
RTN01
Pop ICU Status from BIU Stack into the ICU Status Register
RTN02
RTN03
Pop ICU Interrupt Mask from BIU Stack into the ICU Mask Register
RTN04
RTN05
Pop Return Address from the MC Stack into the MC Program Counter
RTN05
Continue

Fig. B-4.A. Functional Flowchart for Return Routine (RTN)

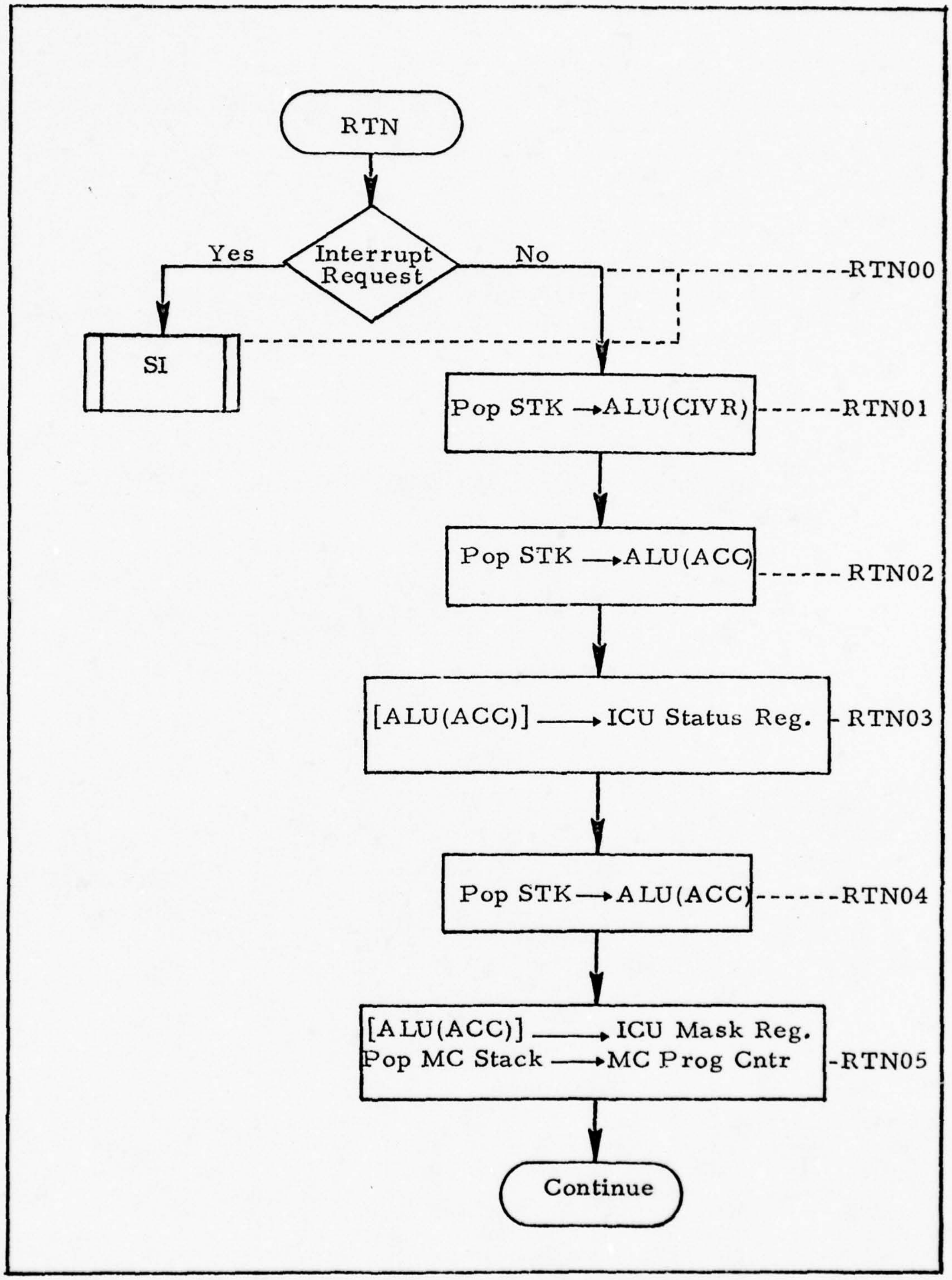


Fig. B-4.B. RTL for Return Routine

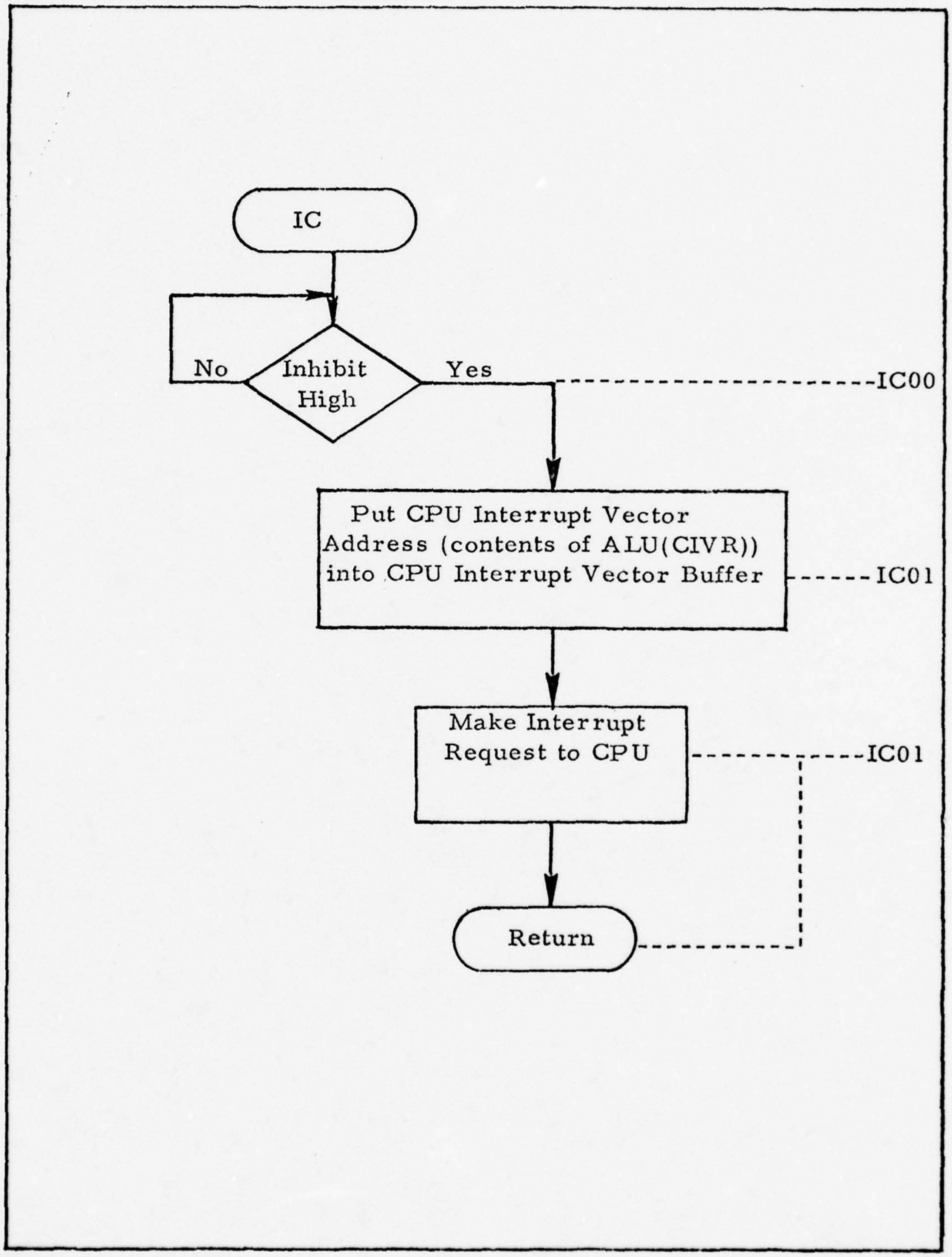

IC
No
Inhibit High
Yes
IC00
Put CPU Interrupt Vector Address (contents of ALU(CIVR)) into CPU Interrupt Vector Buffer
IC01
Make Interrupt Request to CPU
IC01
Return

Fig. B-5.A. Functional Flowchart for Interrupt CPU Routine (IC)

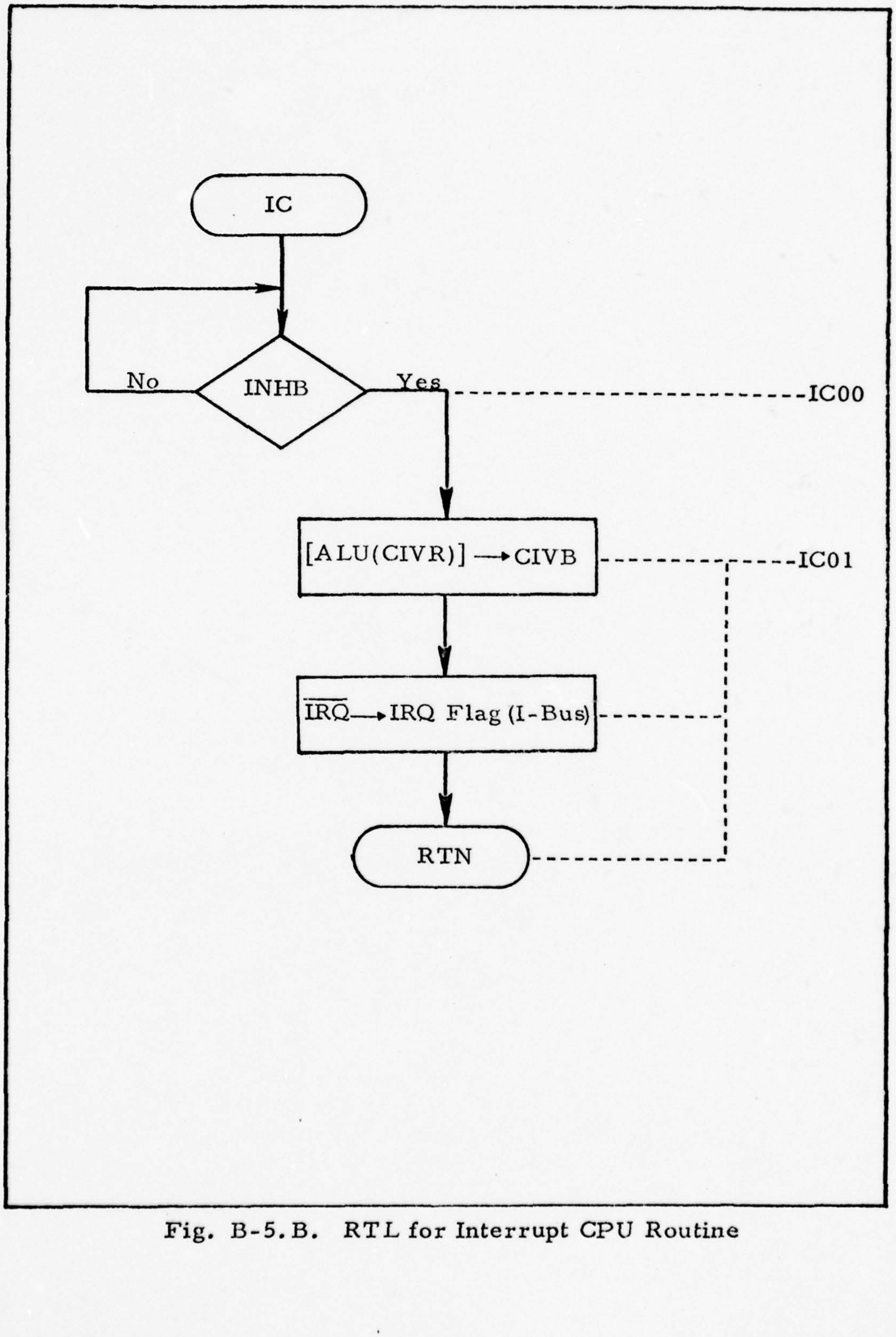
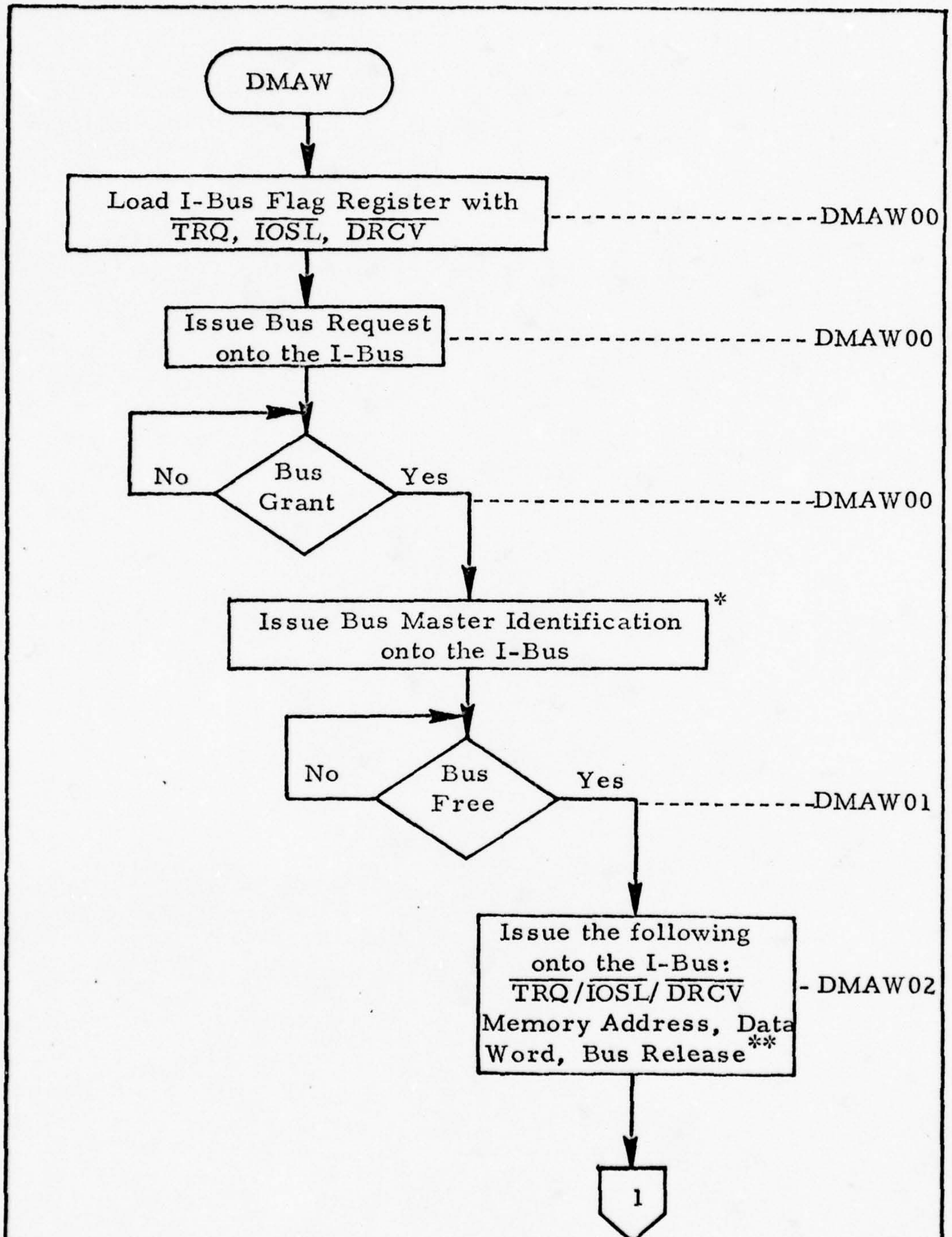


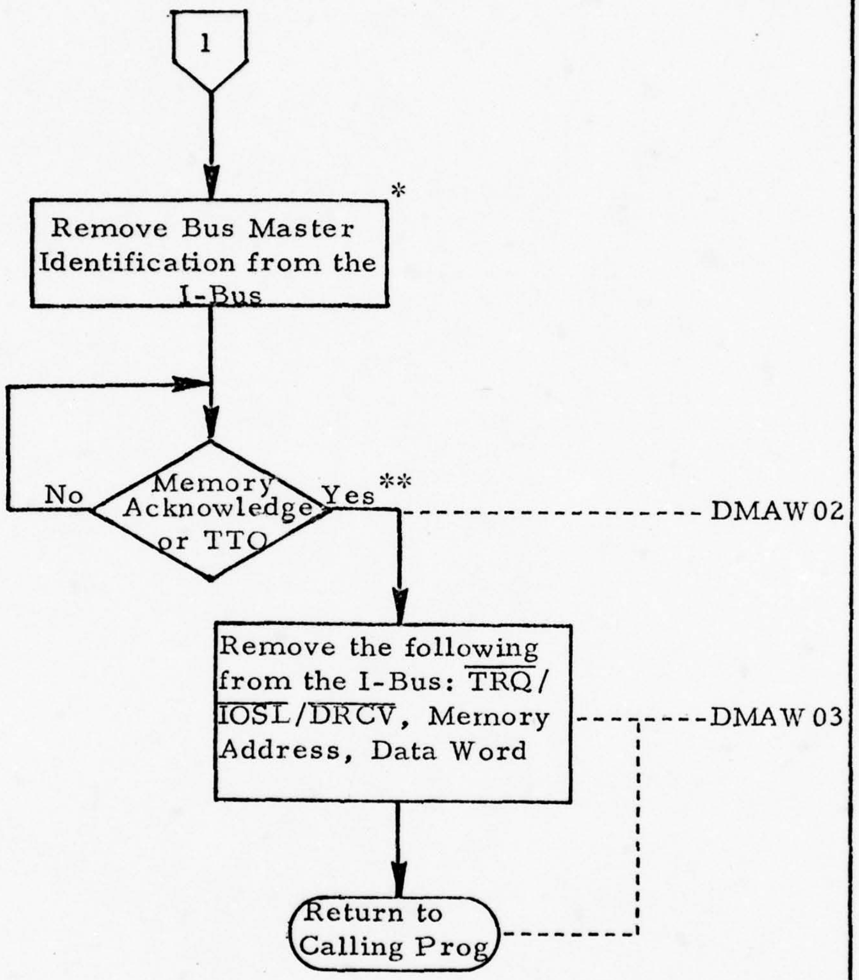
Fig. B-5.B. RTL for Interrupt CPU Routine


DMAW
Load I-Bus Flag Register with $\overline{TRQ}$, $\overline{IOSL}$, $\overline{DRCV}$
DMAW00
Issue Bus Request onto the I-Bus
DMAW00
Bus Grant
No
Yes
DMAW00
Issue Bus Master Identification onto the I-Bus
*
Bus Free
No
Yes
DMAW01
Issue the following onto the I-Bus: $\overline{TRQ}$/$\overline{IOSL}$/$\overline{DRCV}$ Memory Address, Data Word, Bus Release**
DMAW02
1

*Bus Master Identification is issued by hardware upon receipt of Bus Grant.

**Bus Release is issued by hardware when $\overline{TRQ}$ is issued.

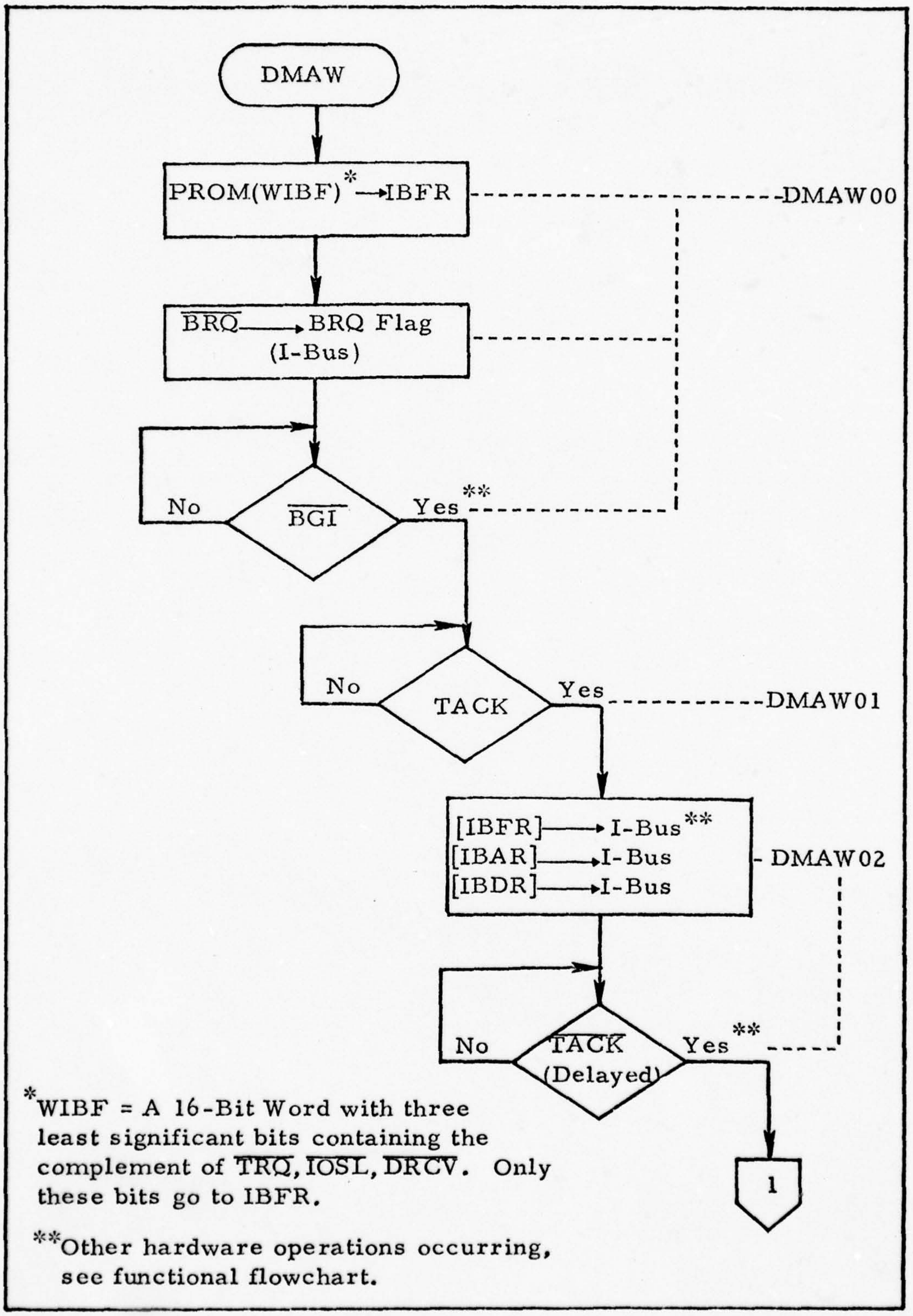
Fig. B-6.A. Functional Flowchart for Direct Memory Access Write Routine (DMAW)


1
Remove Bus Master Identification from the I-Bus
*
No
Memory Acknowledge or TTO
Yes
**
DMAW 02
Remove the following from the I-Bus: $\overline{\text{TRQ}}$/$\overline{\text{IOSL}}$/$\overline{\text{DRCV}}$, Memory Address, Data Word
DMAW 03
Return to Calling Prog

*Bus Master Identification is removed by hardware when Bus Release is issued.

**If TTO occurs, the YES condition will be met. The TTO Interrupt will be set and will be detected at the next check for Interrupt.

Fig. B-6.A. (continued)



*WIBF = A 16-Bit Word with three least significant bits containing the complement of $\overline{TRQ}$, $\overline{IOSL}$, $\overline{DRCV}$. Only these bits go to IBFR.

**Other hardware operations occurring, see functional flowchart.

Fig. B-6.B. RTL for Direct Memory Access Write Routine

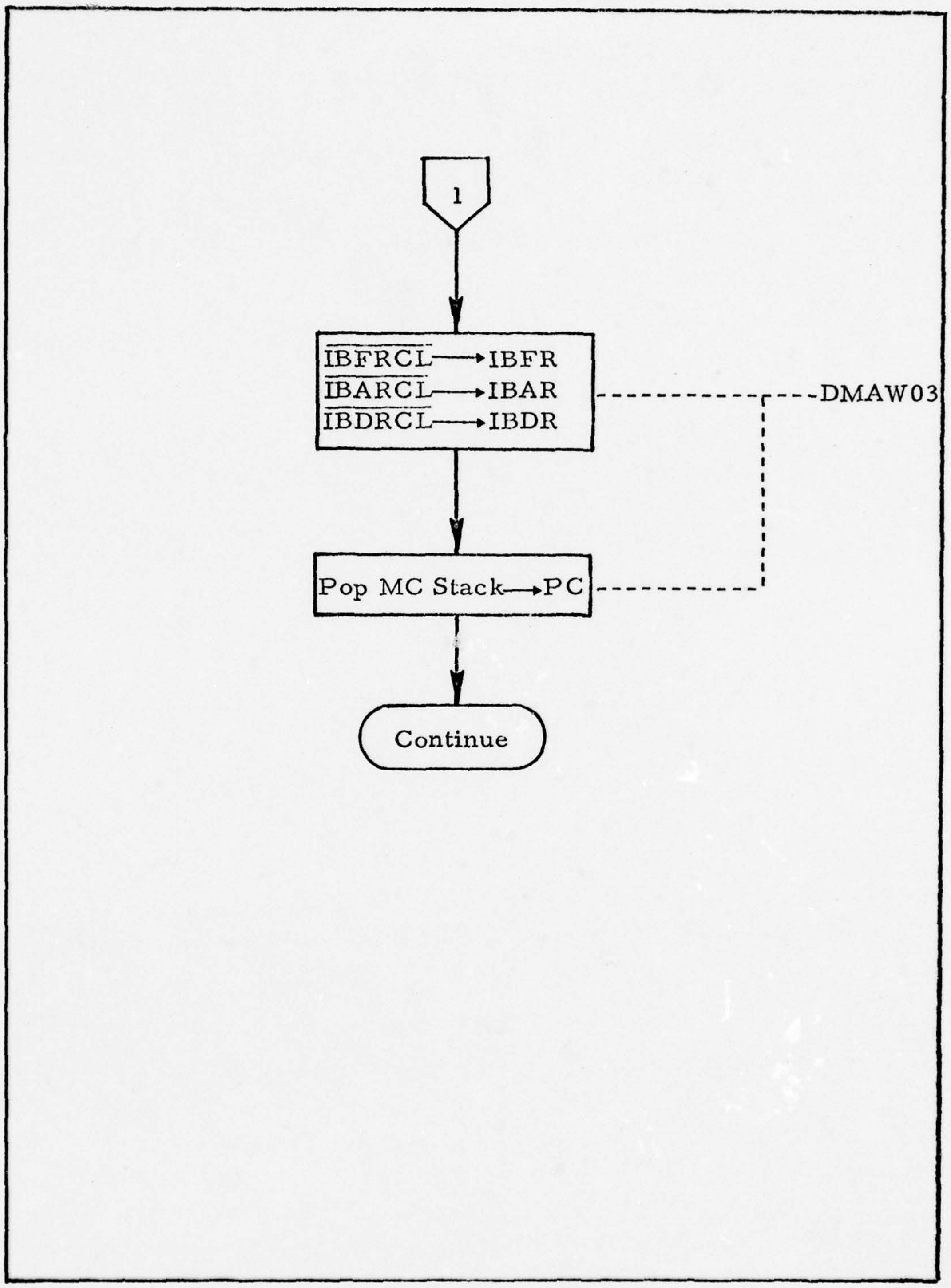

1
$\overline{\text{IBFRCL}} \rightarrow$ IBFR
$\overline{\text{IBARCL}} \rightarrow$ IBAR
$\overline{\text{IBDRCL}} \rightarrow$ IBDR
DMAW03
Pop MC Stack → PC
Continue

Fig. B-6. B. (continued)

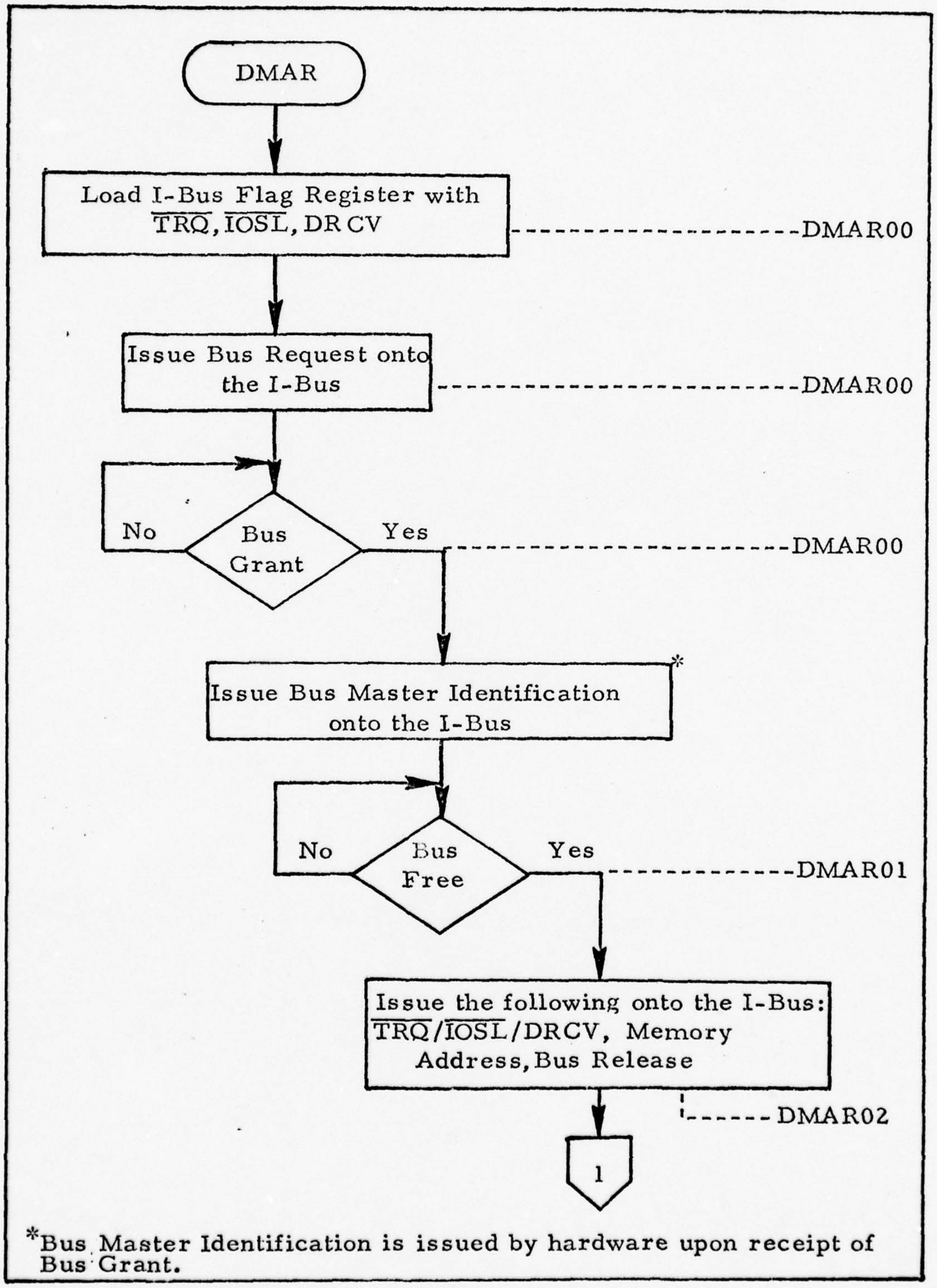
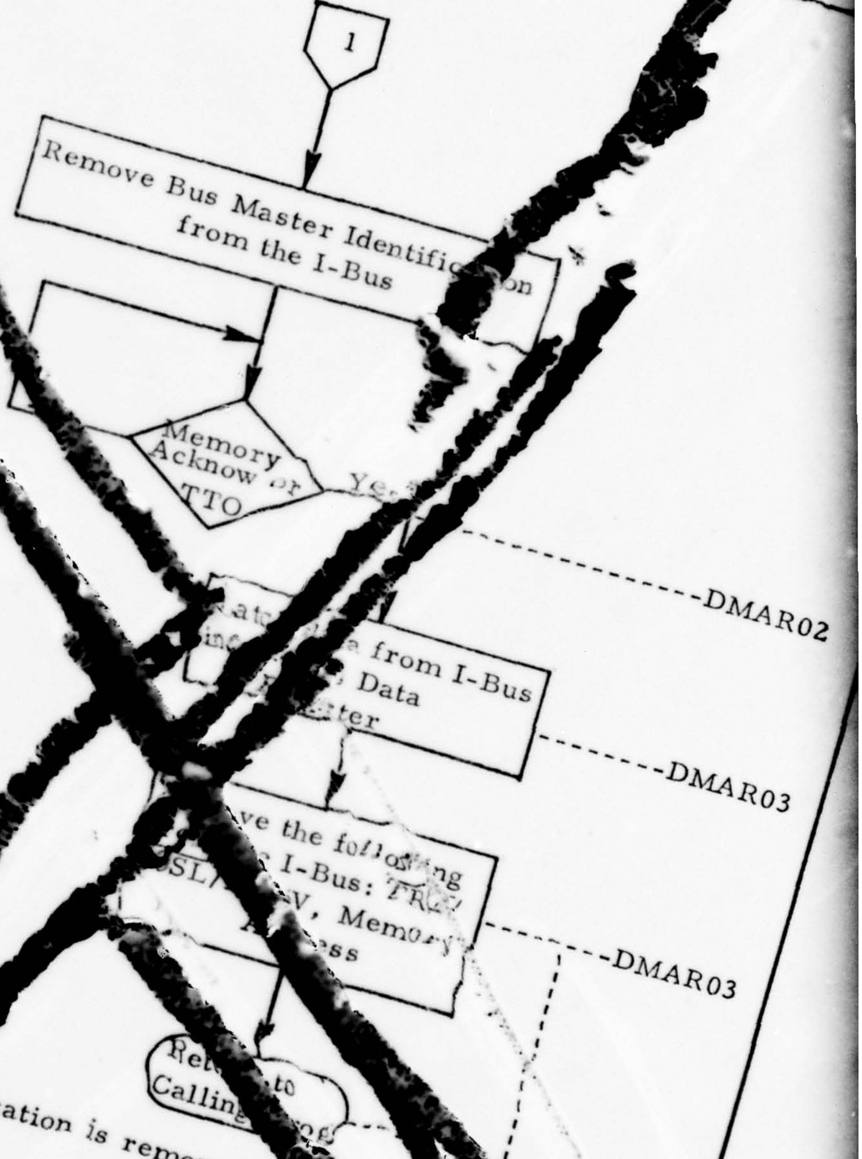

DMAR
Load I-Bus Flag Register with $\overline{TRQ}$, $\overline{IOSL}$, DR CV
DMAR00
Issue Bus Request onto the I-Bus
DMAR00
No
Bus Grant
Yes
DMAR00
Issue Bus Master Identification onto the I-Bus
*
No
Bus Free
Yes
DMAR01
Issue the following onto the I-Bus: $\overline{TRQ}$/$\overline{IOSL}$/DRCV, Memory Address, Bus Release
DMAR02
1
*Bus Master Identification is issued by hardware upon receipt of Bus Grant.

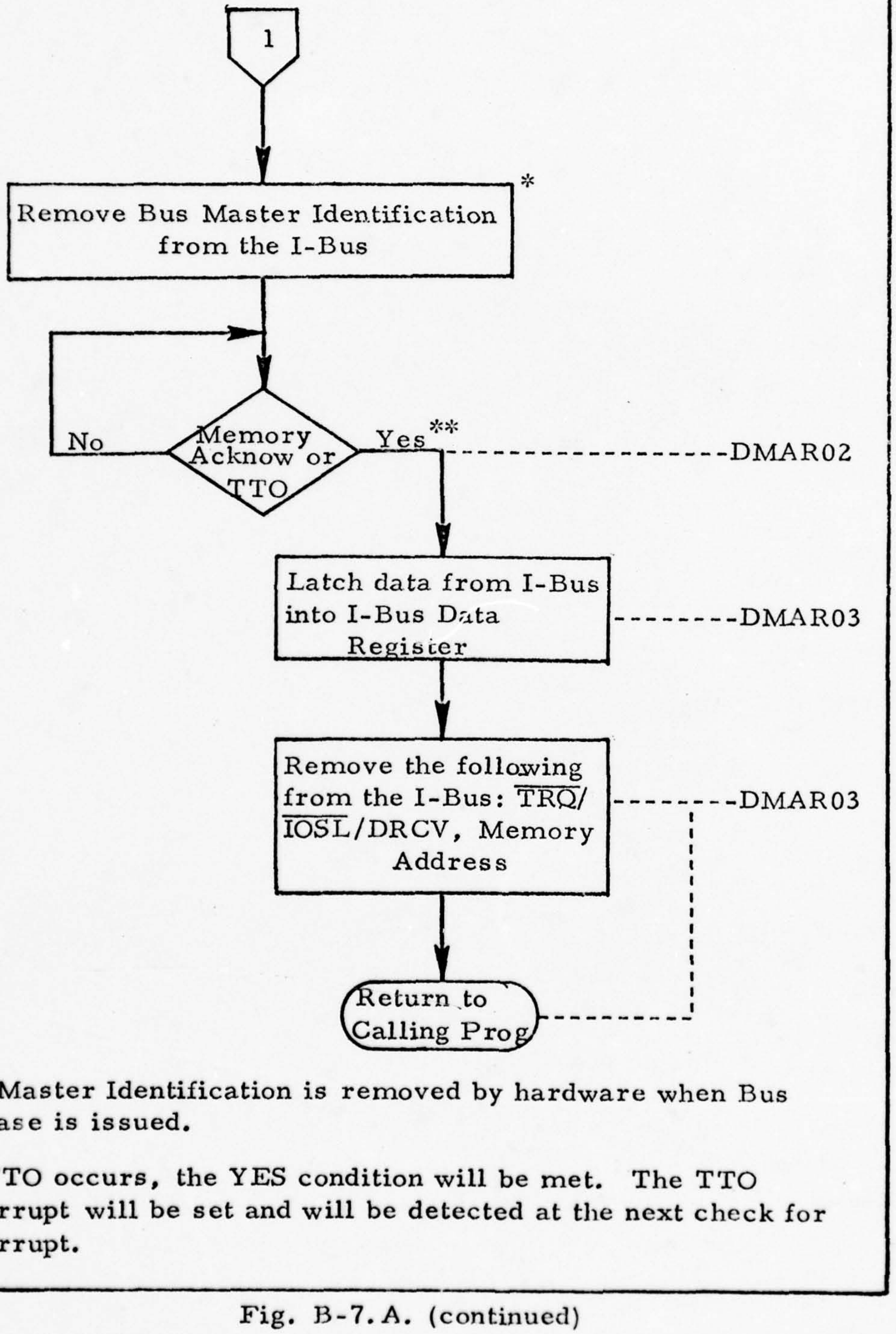
Fig. B-7.A. Functional Flowchart for Direct Memory Access Read Routine (DMAR)


1
Remove Bus Master Identification from the I-Bus
Memory Acknow or TTO
Yes
Latch data from I-Bus into the Data Register
DMAR02
DMAR03
Remove the following from the I-Bus: TRQ, SL/..., Memory Address
DMAR03
Return to Calling Prog

~~*Bus Master Identification is removed by hardware when Release is issued.~~

~~**If TTO occurs, the YES condition will be met. The TTO Interrupt will be set and will be detected at the next check for Interrupt.~~

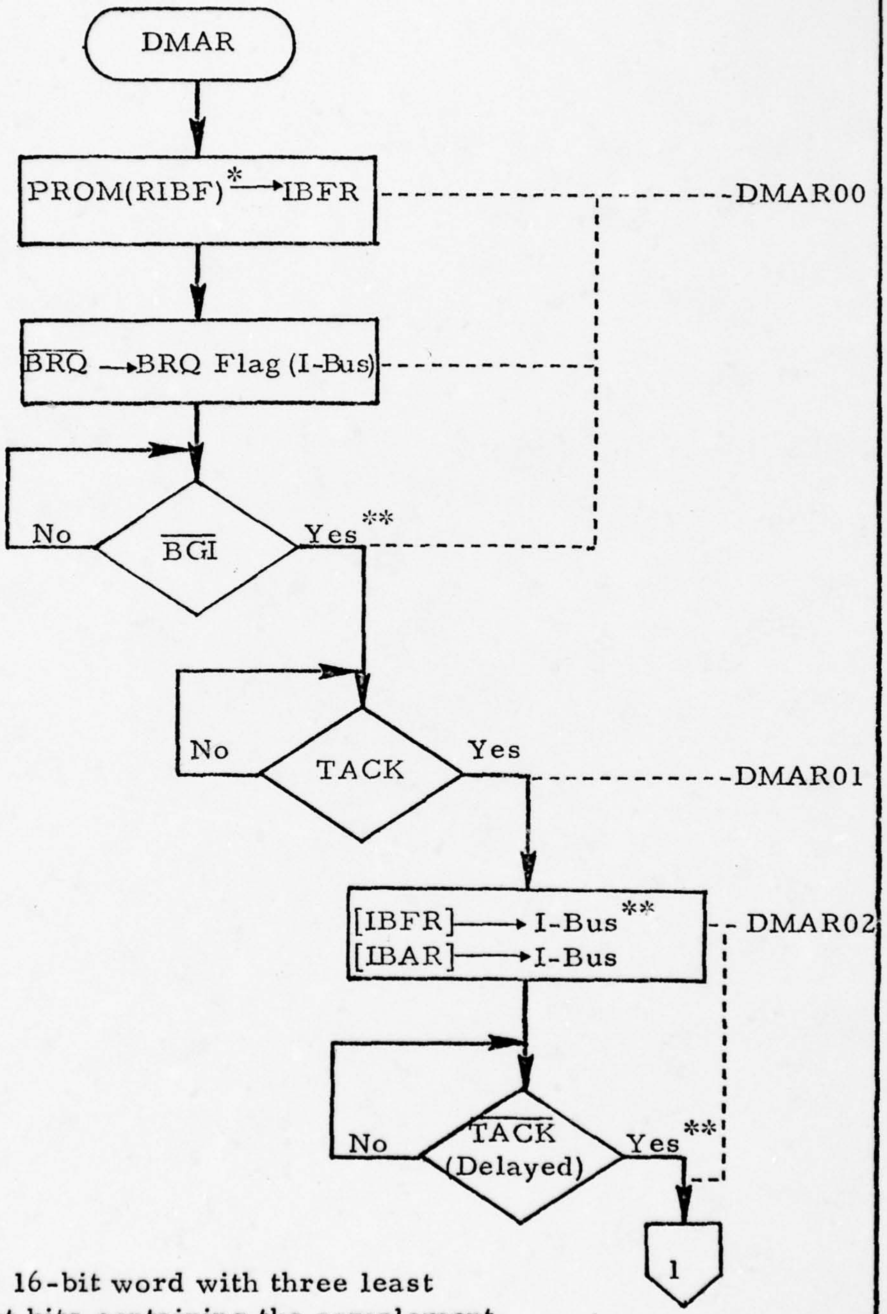
~~Fig. B-7.A. (continued)~~


1
Remove Bus Master Identification from the I-Bus
*
Memory Acknow or TTO
No
Yes
**
DMAR02
Latch data from I-Bus into I-Bus Data Register
DMAR03
Remove the following from the I-Bus: $\overline{TRQ}$/ $\overline{IOSL}$/DRCV, Memory Address
DMAR03
Return to Calling Prog

*Bus Master Identification is removed by hardware when Bus Release is issued.

**If TTO occurs, the YES condition will be met. The TTO Interrupt will be set and will be detected at the next check for Interrupt.

Fig. B-7.A. (continued)



*RIBF = A 16-bit word with three least significant bits containing the complement of $\overline{TRQ}$, $\overline{IOSL}$, DRCV. Only these bits go to IBFR.

**Other hardware operations occurring, see functional flowchart.

Fig. B-7.B. RTL for Direct Memory Access Read Routine

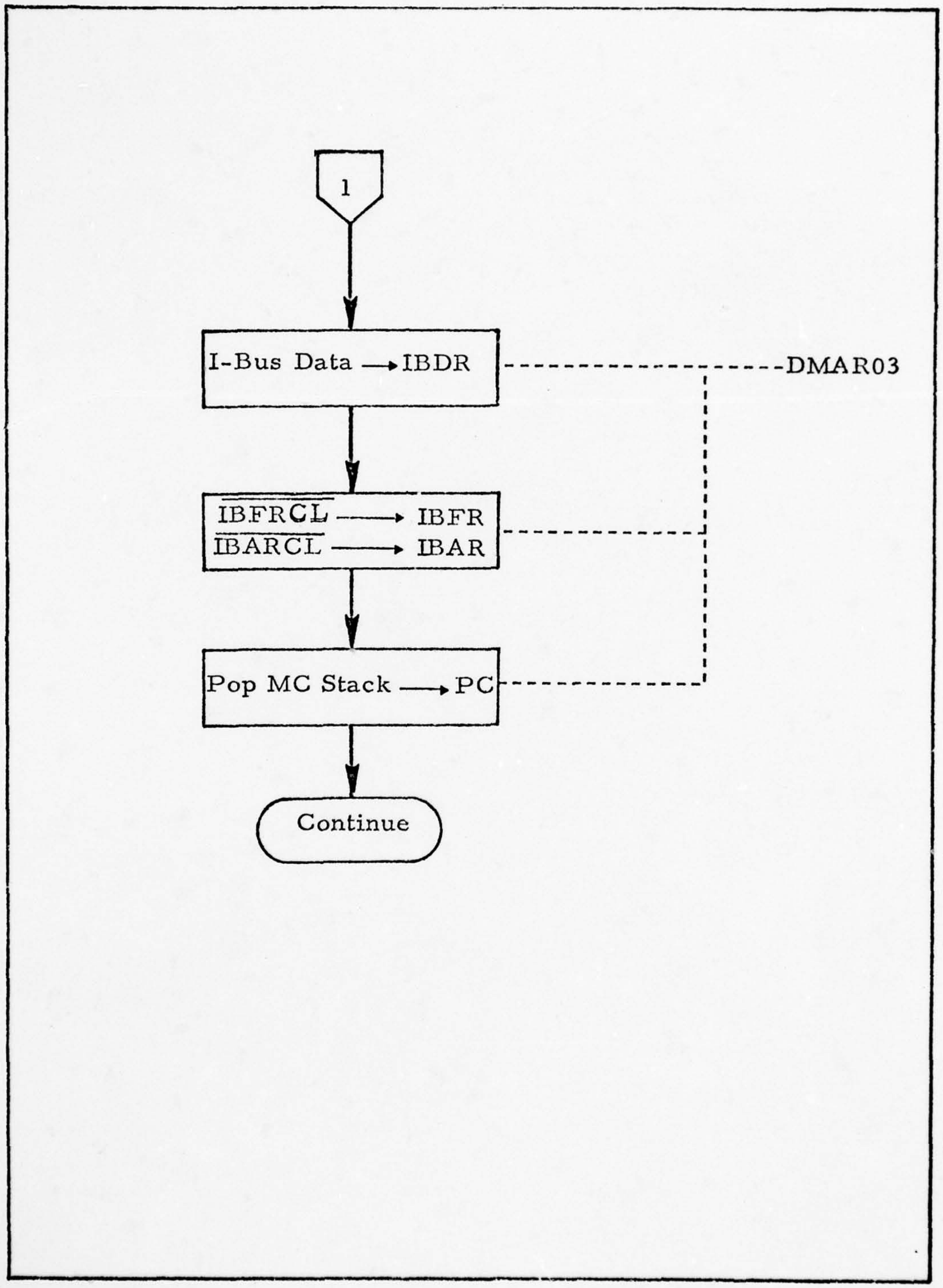

1
I-Bus Data → IBDR
DMAR03
$\overline{\text{IBFRCL}}$ → IBFR
$\overline{\text{IBARCL}}$ → IBAR
Pop MC Stack → PC
Continue

Fig. B-7.B. (continued)

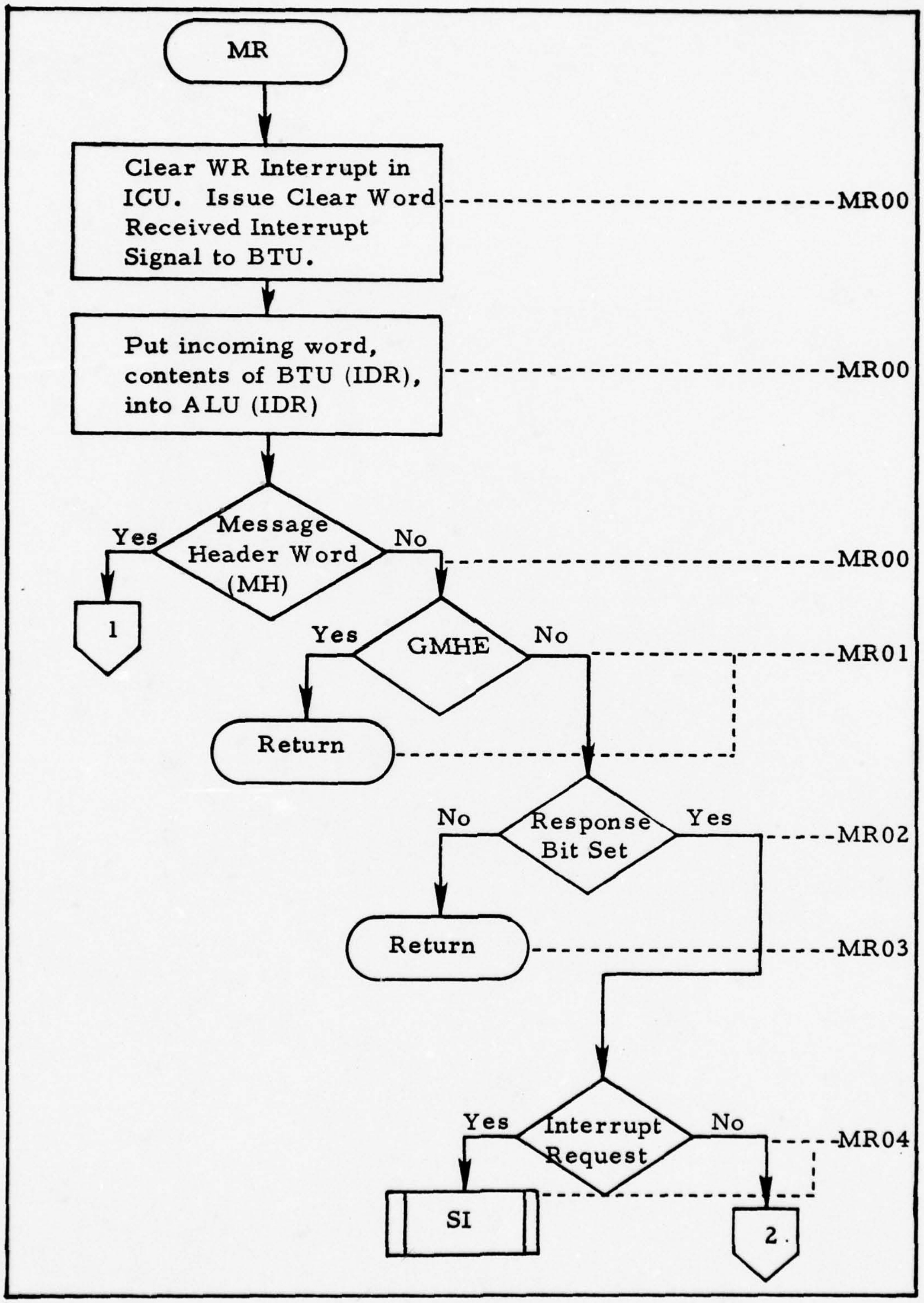

MR
Clear WR Interrupt in ICU. Issue Clear Word Received Interrupt Signal to BTU.
MR00
Put incoming word, contents of BTU (IDR), into ALU (IDR)
MR00
Message Header Word (MH)
Yes
No
MR00
1
GMHE
Yes
No
MR01
Return
Response Bit Set
No
Yes
MR02
Return
MR03
Interrupt Request
Yes
No
MR04
SI
2

Fig. B-8.A. Functional Flowchart for Message Reception Routine (MR)

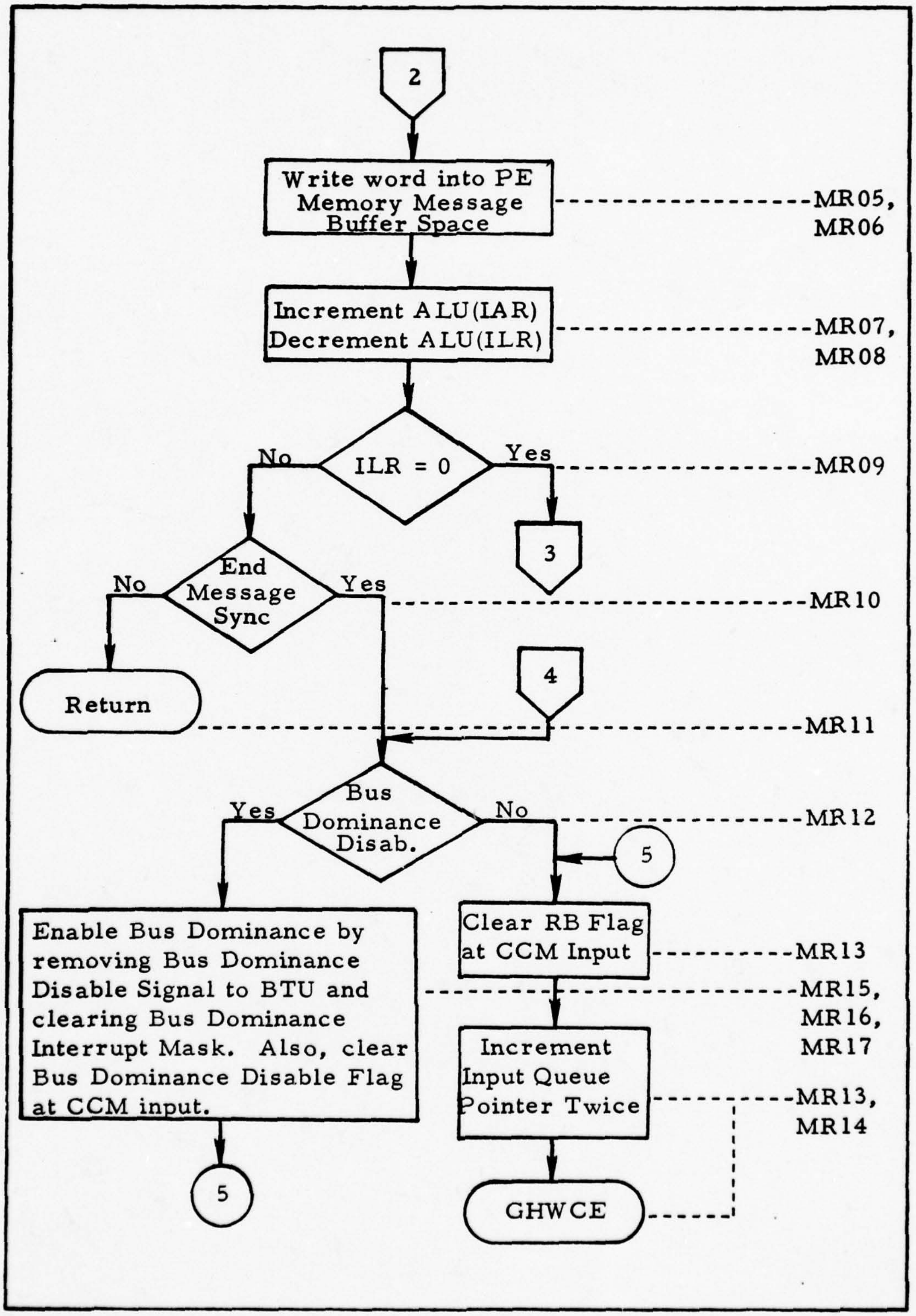

2
Write word into PE Memory Message Buffer Space
MR05, MR06
Increment ALU(IAR) Decrement ALU(ILR)
MR07, MR08
No
ILR = 0
Yes
MR09
3
No
End Message Sync
Yes
MR10
4
Return
MR11
Yes
Bus Dominance Disab.
No
MR12
5
Enable Bus Dominance by removing Bus Dominance Disable Signal to BTU and clearing Bus Dominance Interrupt Mask. Also, clear Bus Dominance Disable Flag at CCM input.
Clear RB Flag at CCM Input
MR13
MR15, MR16, MR17
Increment Input Queue Pointer Twice
MR13, MR14
5
GHWCE

Fig. B-8.A. (continued)

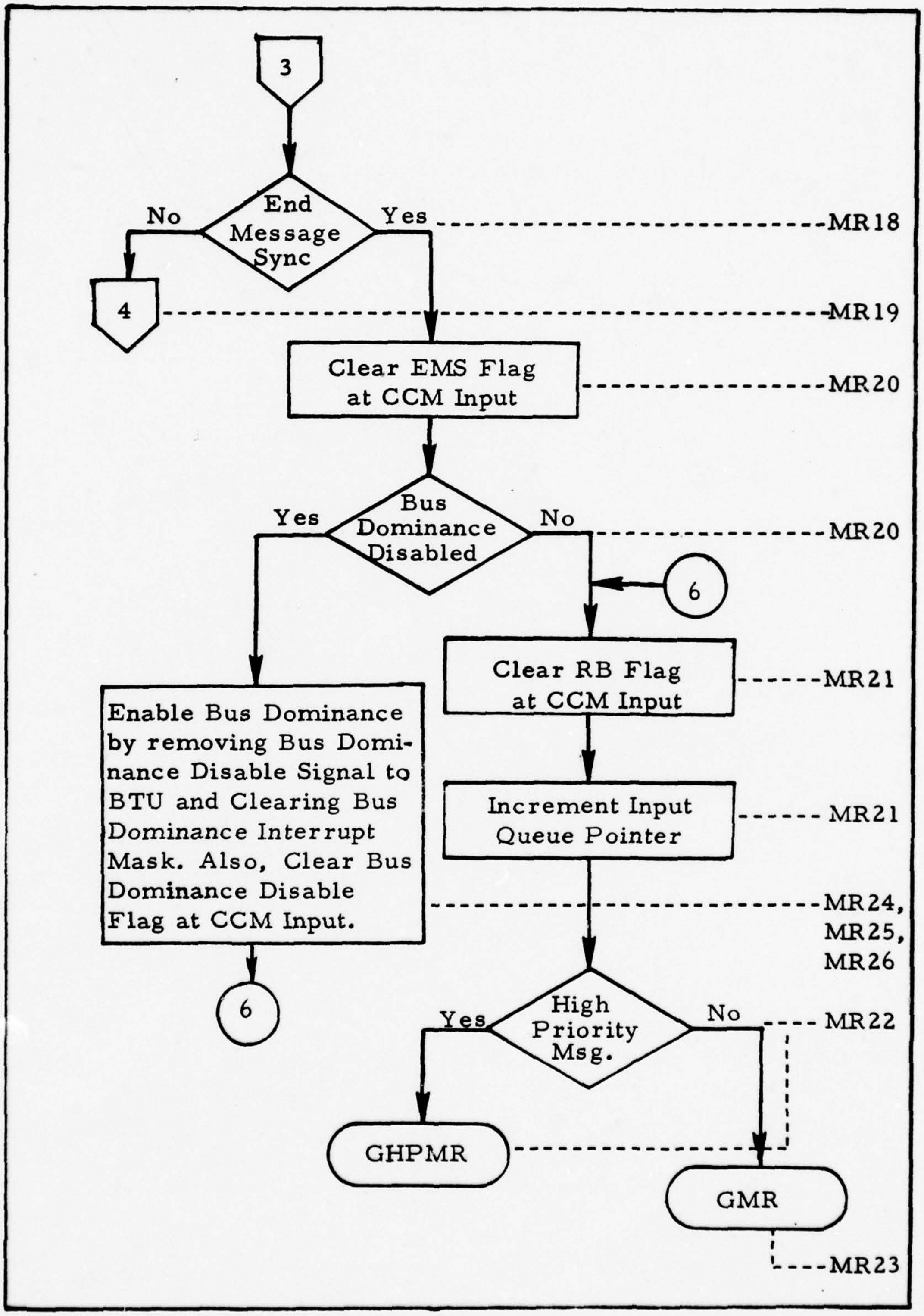

3
End Message Sync
No
Yes
MR18
4
MR19
Clear EMS Flag at CCM Input
MR20
Bus Dominance Disabled
Yes
No
MR20
6
Clear RB Flag at CCM Input
MR21
Increment Input Queue Pointer
MR21
Enable Bus Dominance by removing Bus Dominance Disable Signal to BTU and Clearing Bus Dominance Interrupt Mask. Also, Clear Bus Dominance Disable Flag at CCM Input.
MR24,
MR25,
MR26
6
High Priority Msg.
Yes
No
MR22
GHPMR
GMR
MR23

Fig. B-8. A. (continued)

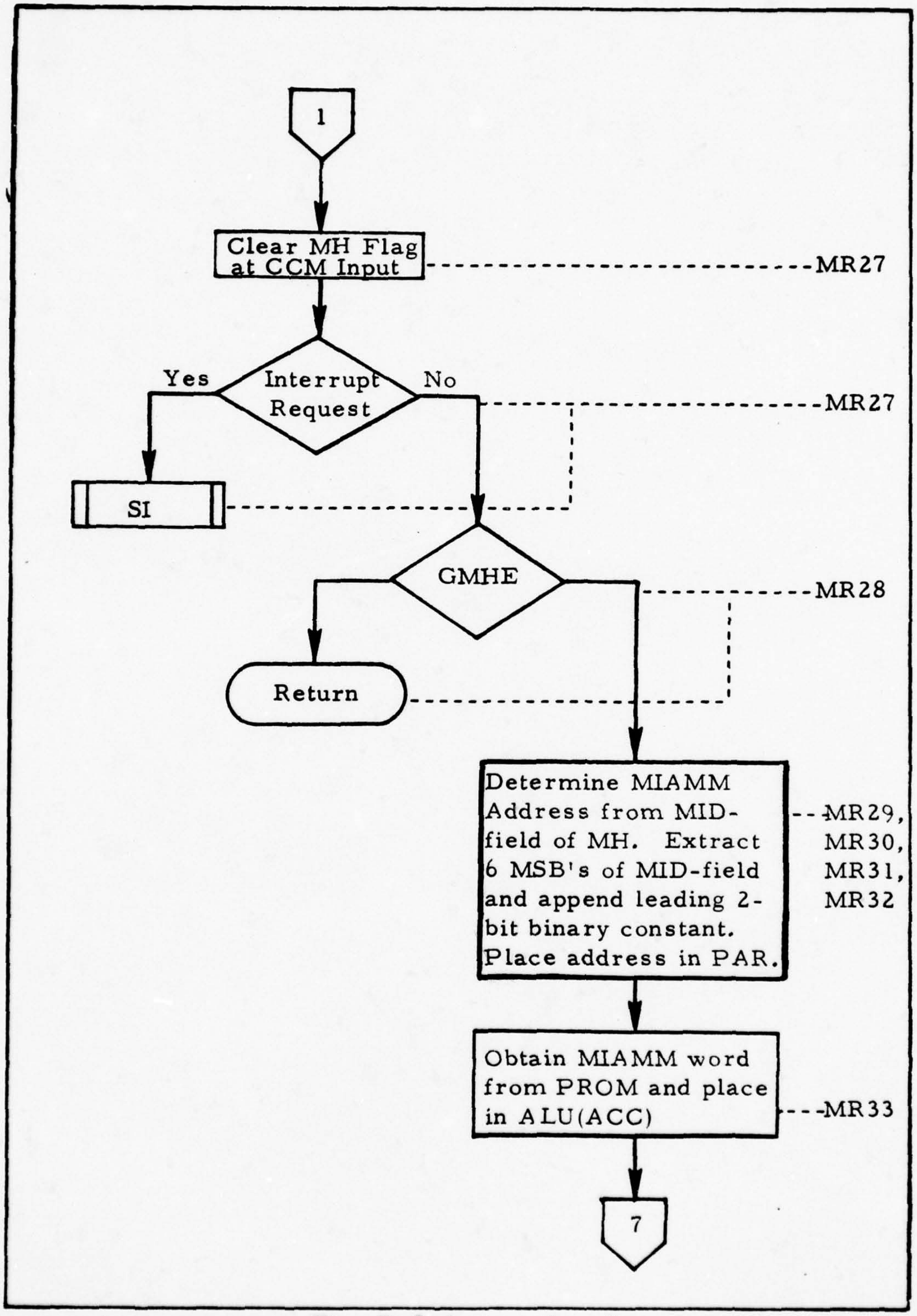

1
Clear MH Flag at CCM Input
MR27
Interrupt Request
Yes
No
MR27
SI
GMHE
MR28
Return
Determine MIAMM Address from MID-field of MH. Extract 6 MSB's of MID-field and append leading 2-bit binary constant. Place address in PAR.
MR29, MR30, MR31, MR32
Obtain MIAMM word from PROM and place in ALU(ACC)
MR33
7

Fig. B-8.A. (continued)

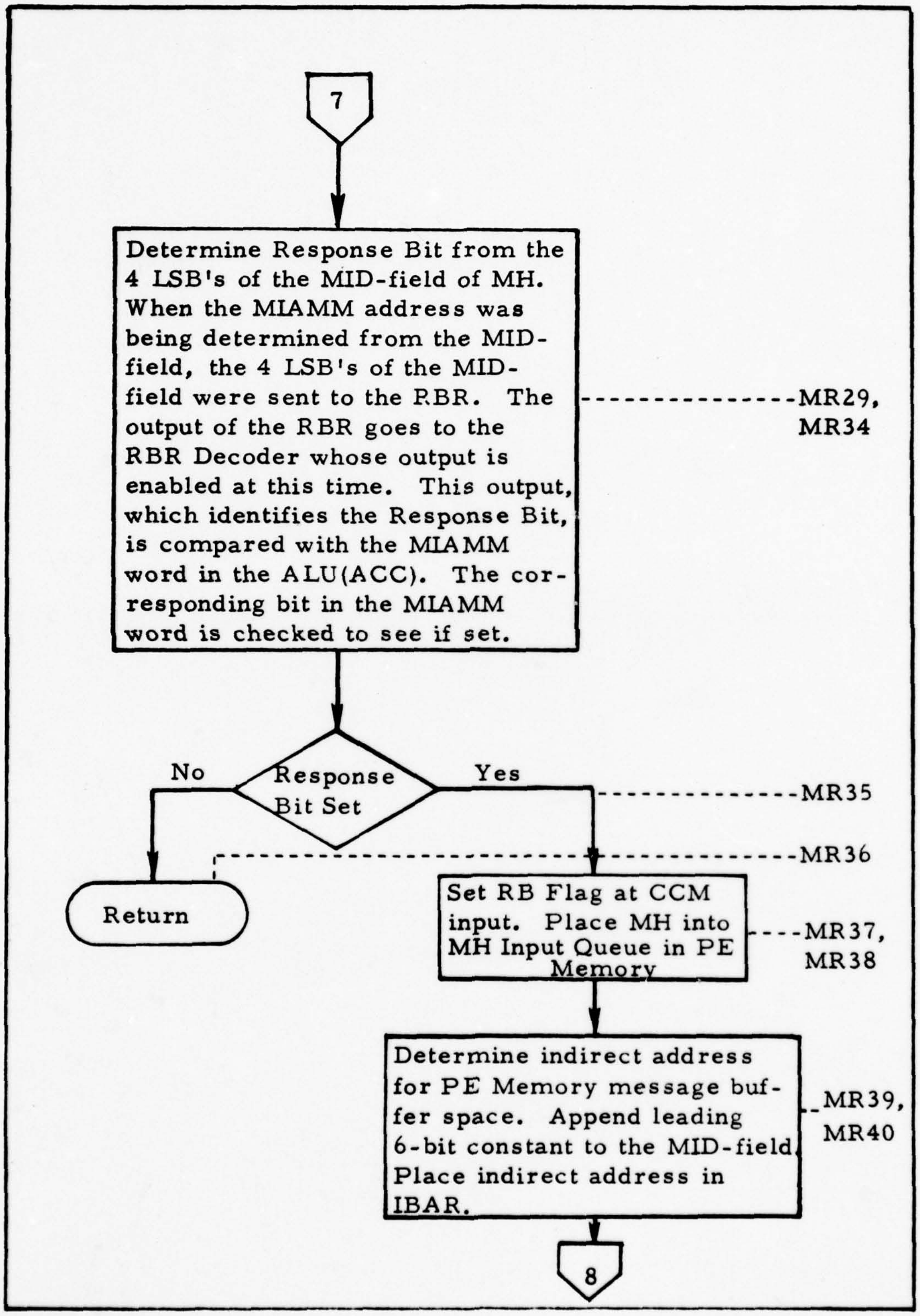

7
Determine Response Bit from the 4 LSB's of the MID-field of MH. When the MIAMM address was being determined from the MID-field, the 4 LSB's of the MID-field were sent to the RBR. The output of the RBR goes to the RBR Decoder whose output is enabled at this time. This output, which identifies the Response Bit, is compared with the MIAMM word in the ALU(ACC). The corresponding bit in the MIAMM word is checked to see if set.
MR29, MR34
Response Bit Set
No
Yes
MR35
MR36
Return
Set RB Flag at CCM input. Place MH into MH Input Queue in PE Memory
MR37, MR38
Determine indirect address for PE Memory message buffer space. Append leading 6-bit constant to the MID-field. Place indirect address in IBAR.
MR39, MR40
8

Fig. B-8.A. (continued)

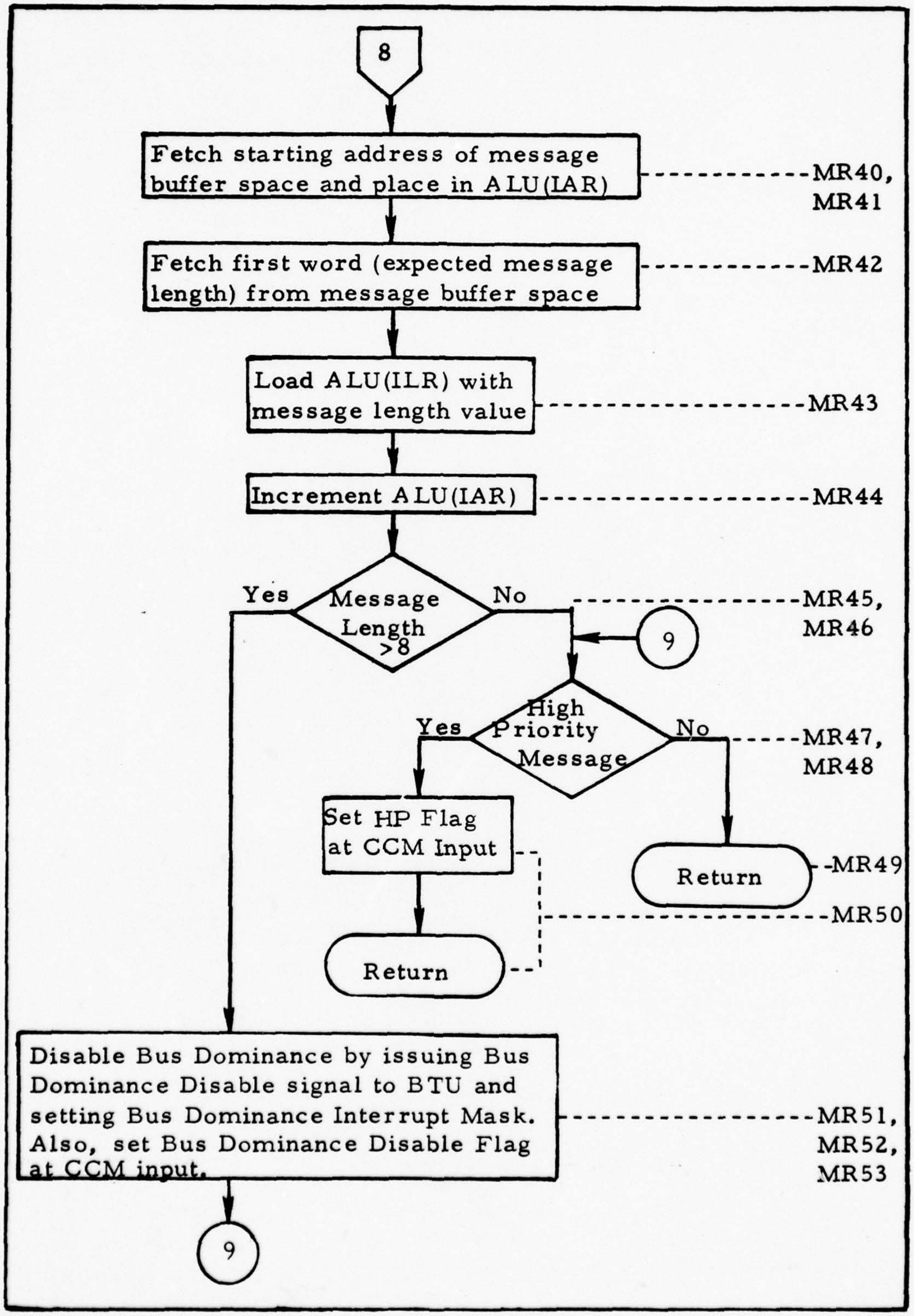

8
Fetch starting address of message buffer space and place in ALU(IAR)
MR40, MR41
Fetch first word (expected message length) from message buffer space
MR42
Load ALU(ILR) with message length value
MR43
Increment ALU(IAR)
MR44
Message Length >8
Yes
No
MR45, MR46
9
High Priority Message
Yes
No
MR47, MR48
Set HP Flag at CCM Input
Return
MR49
MR50
Return
Disable Bus Dominance by issuing Bus Dominance Disable signal to BTU and setting Bus Dominance Interrupt Mask. Also, set Bus Dominance Disable Flag at CCM input.
MR51, MR52, MR53
9

Fig. B-8.A. (continued)

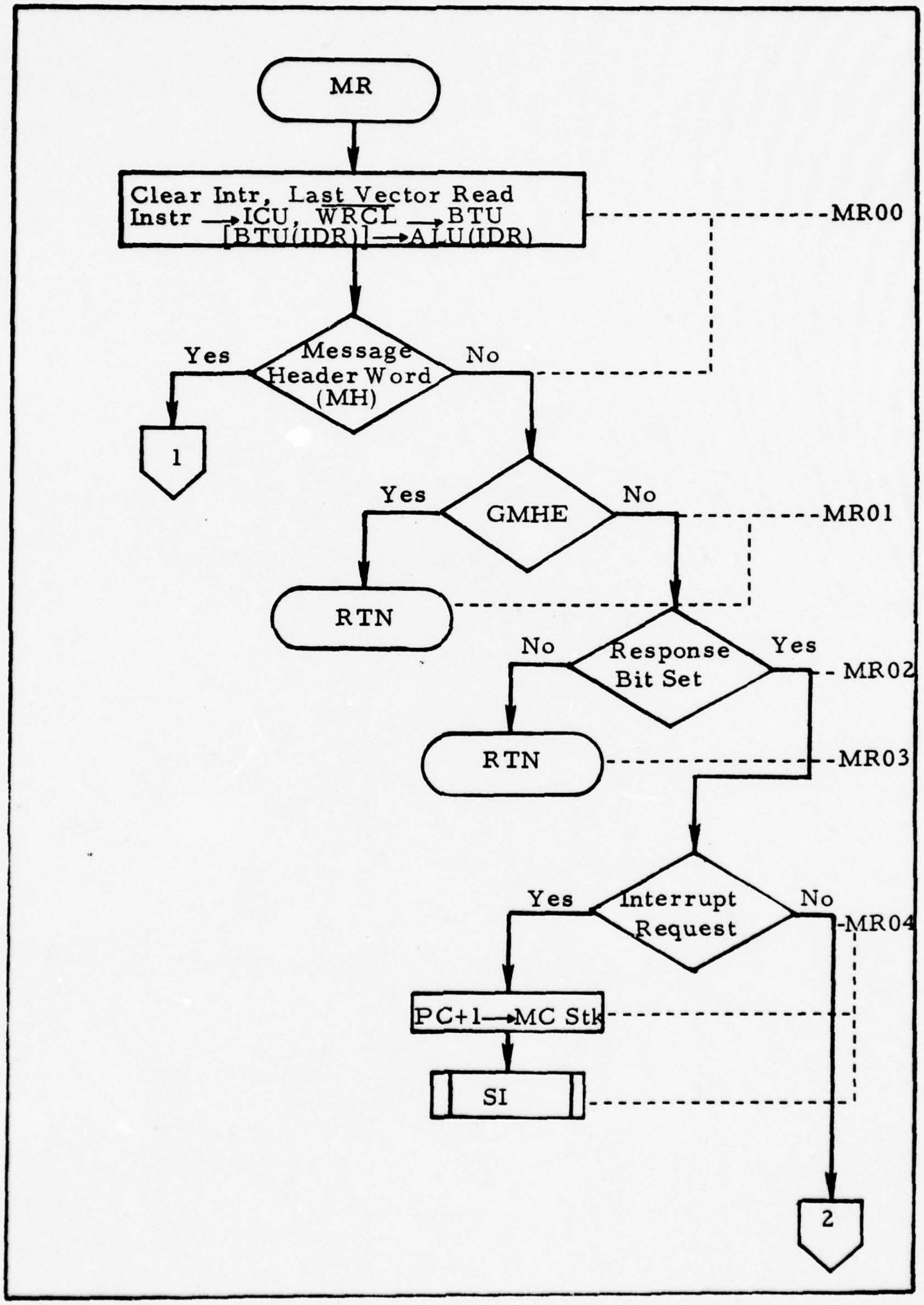


Fig. B-8.B. RTL for Message Reception Routine

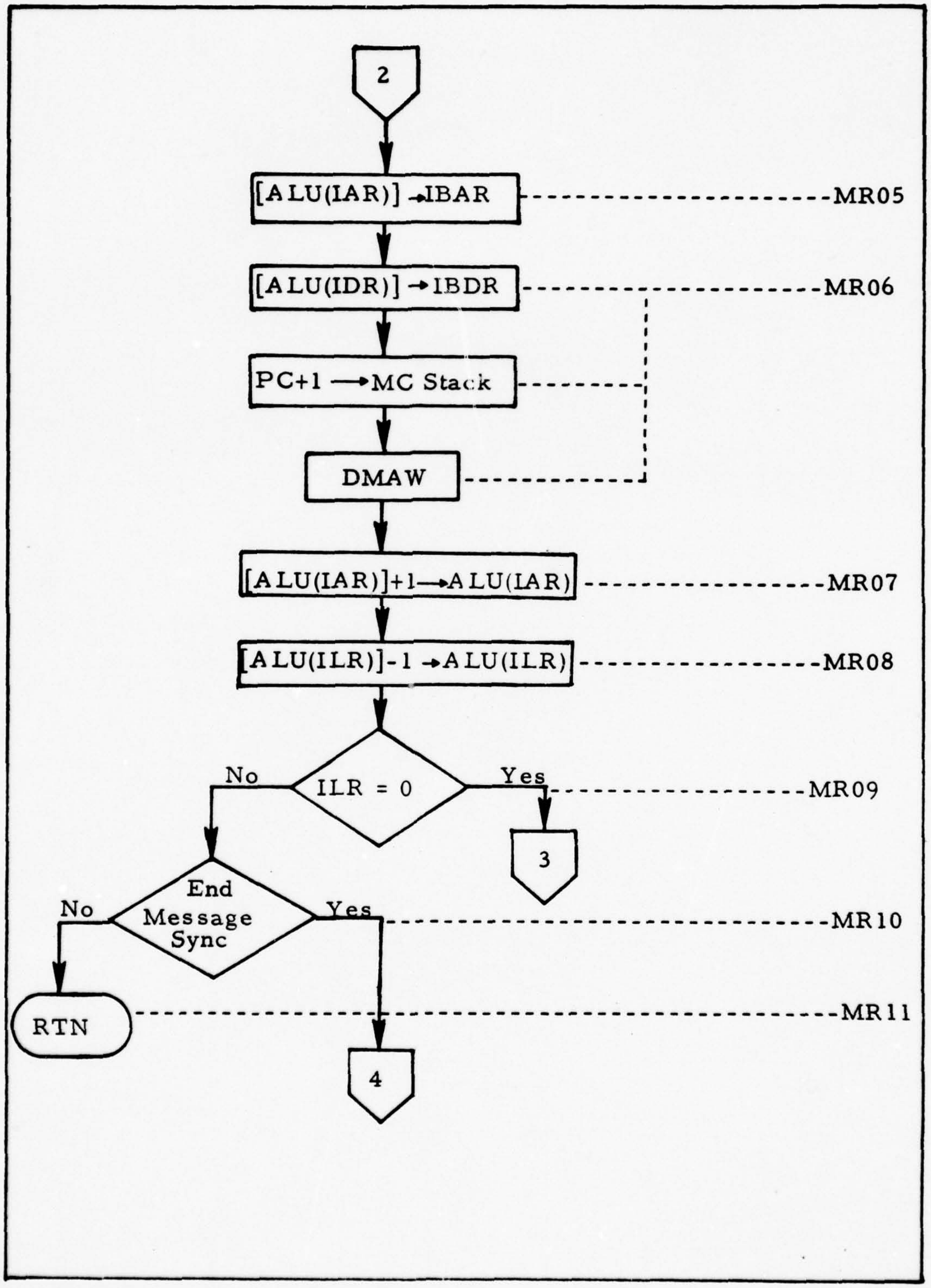
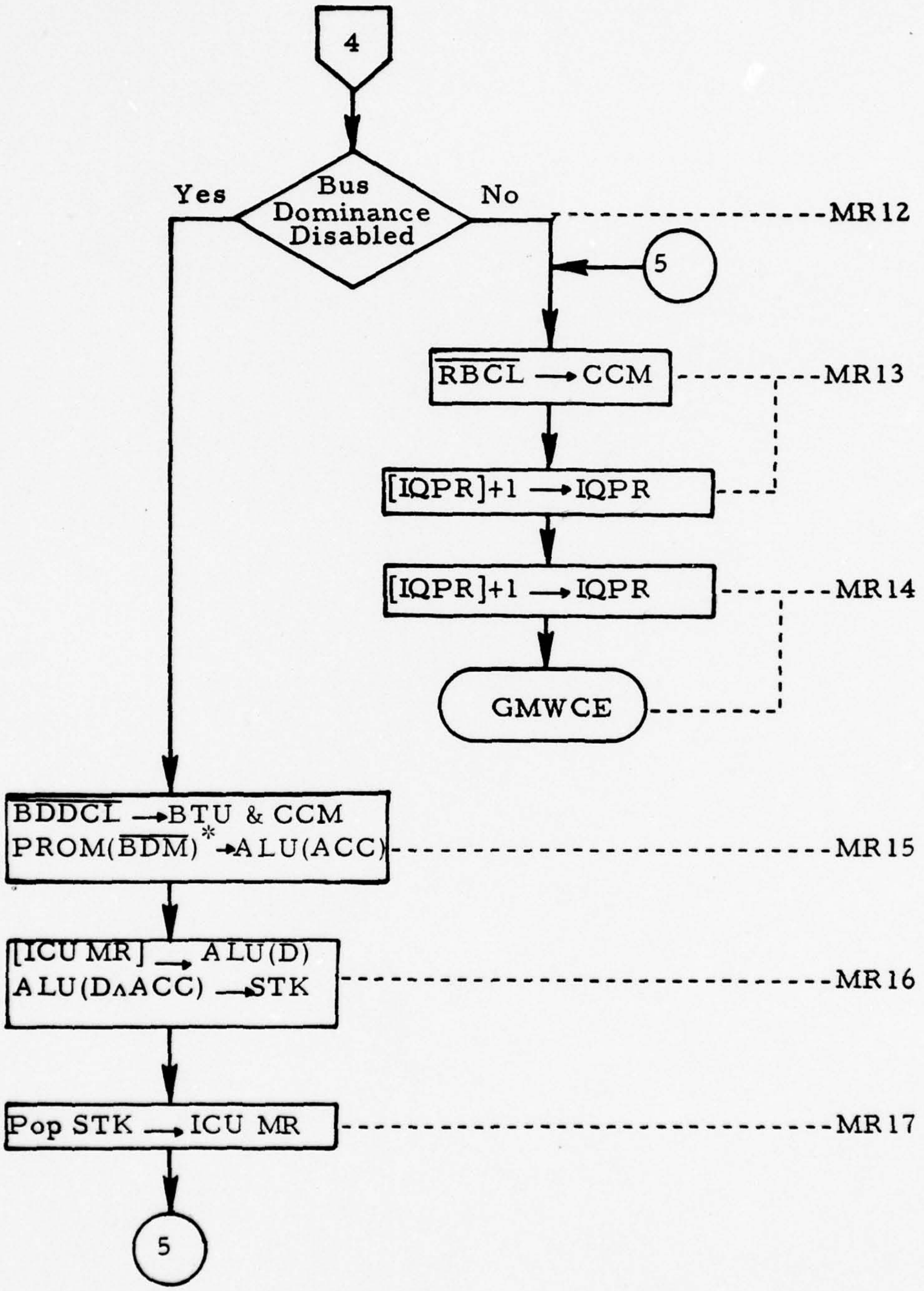


Fig. B-8.B. (continued)



*$\overline{BDM}$ = Clear Bus Dominance Mask word. A 16-bit word with the Bus Dominance bit equal to zero and the other bits equal to one.

Fig. B-8.B. (continued)

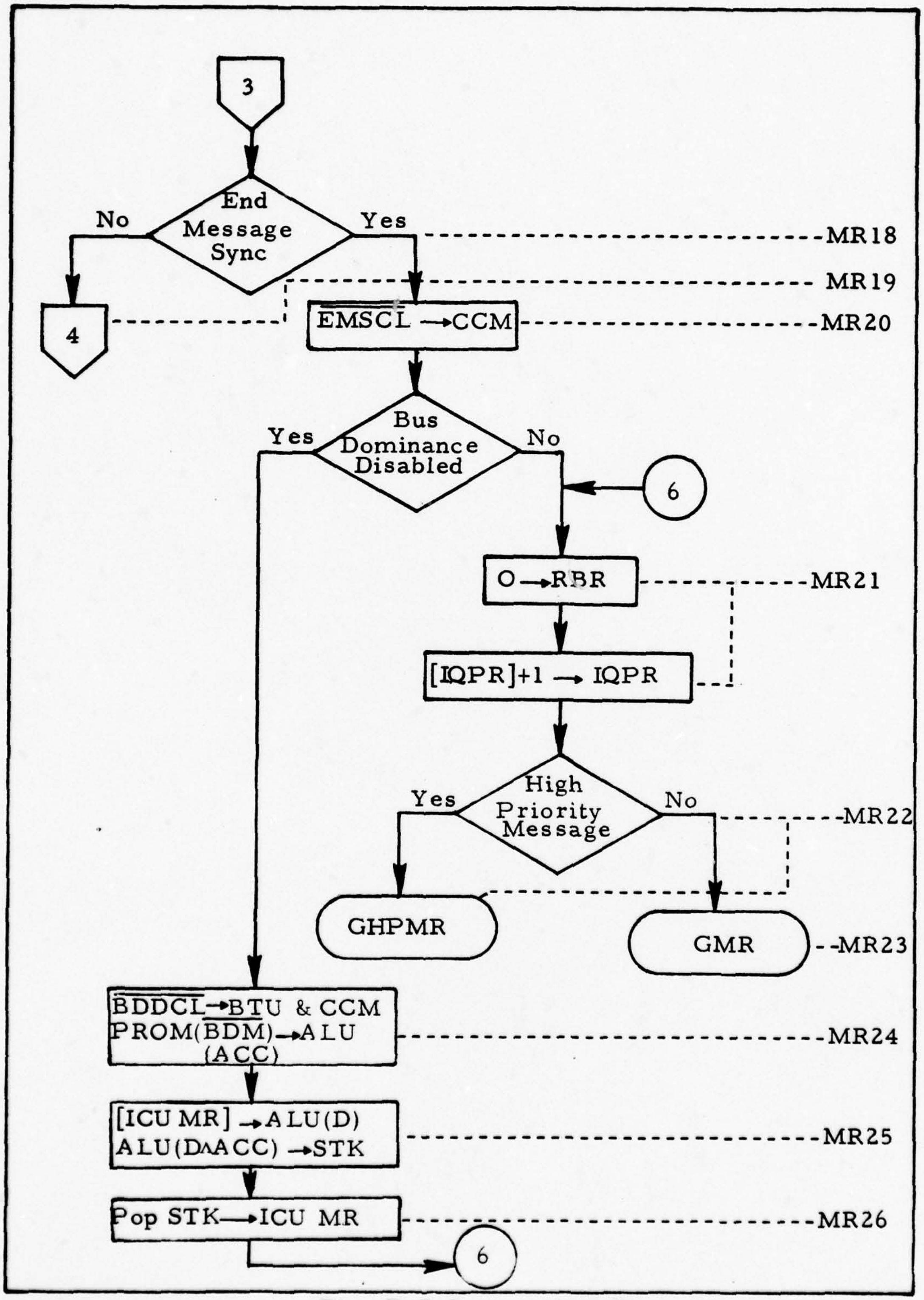
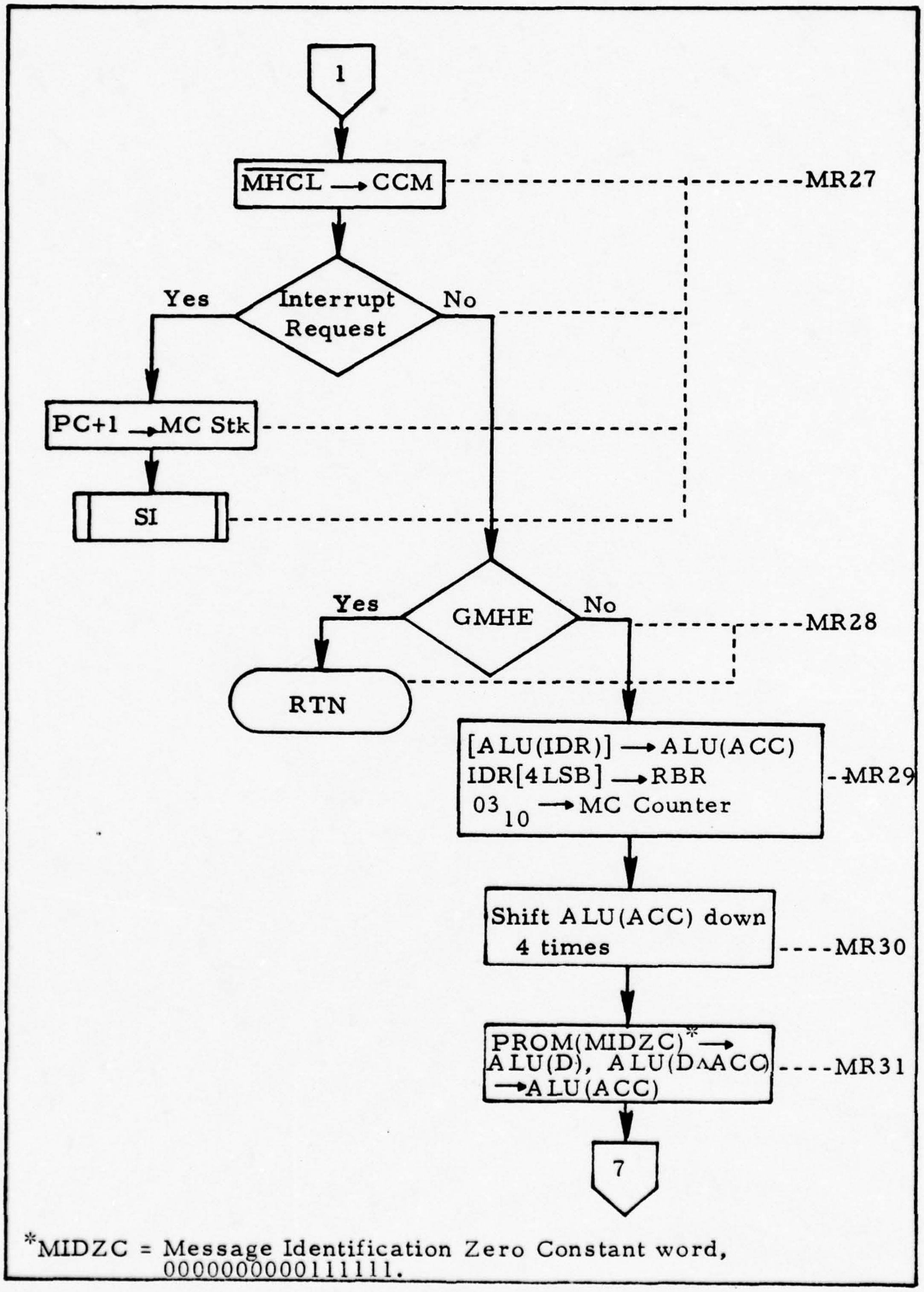
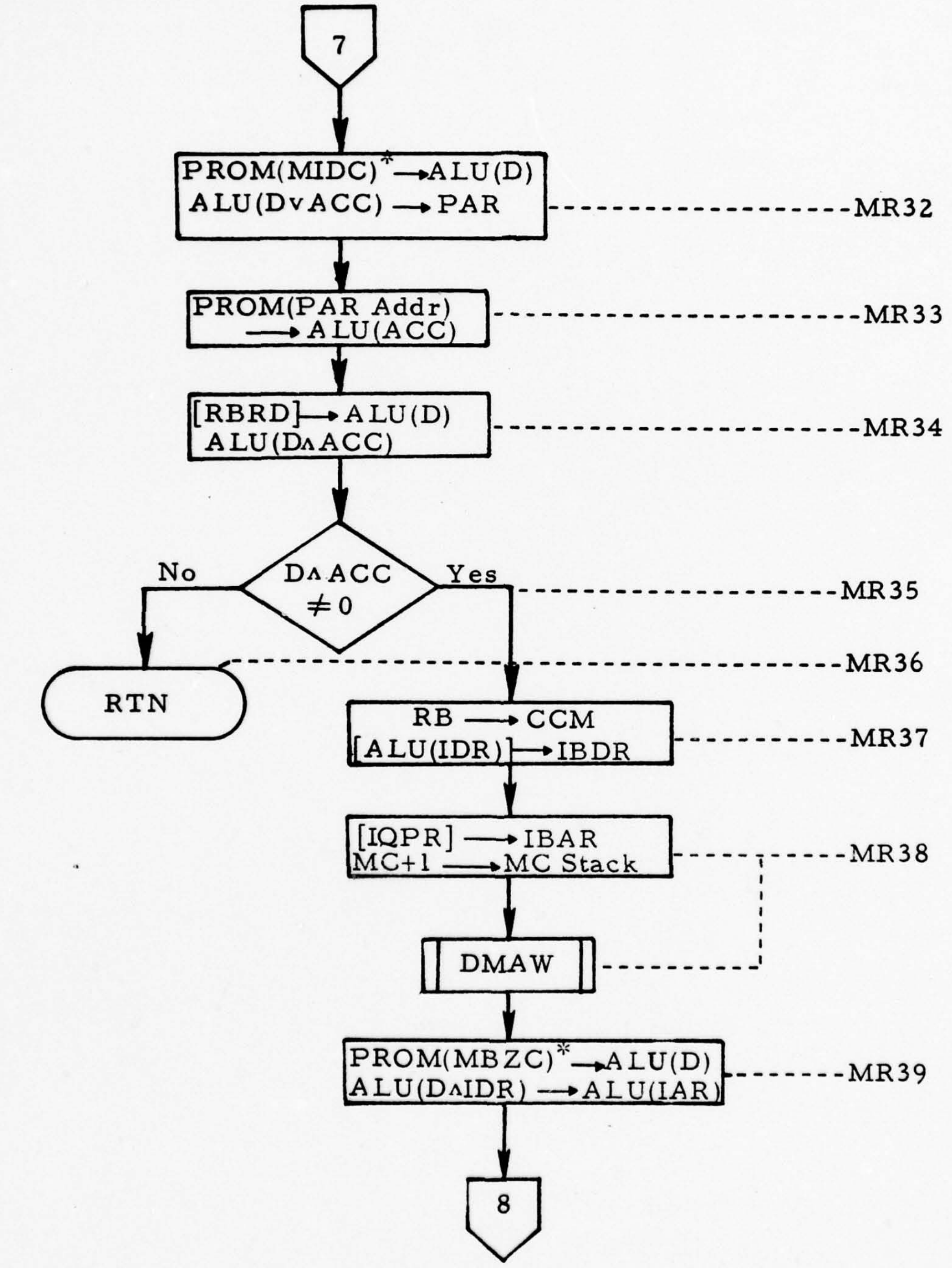


Fig. B-8.B. (continued)



*MIDZC = Message Identification Zero Constant word, 0000000000111111.

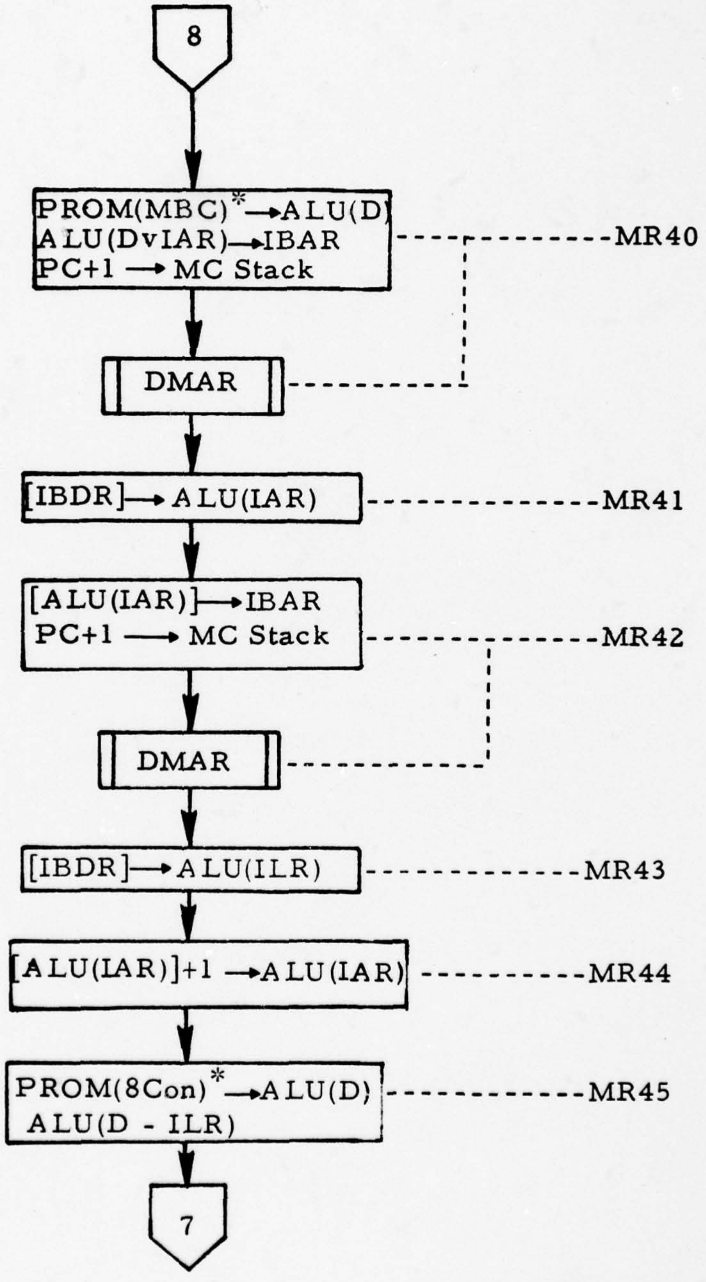
Fig. B-8.B. (continued)



*MIDC = Message Identification Constant, 00000000xx000000

*MBZC = Message Buffer Zero Constant, 0000001111111111

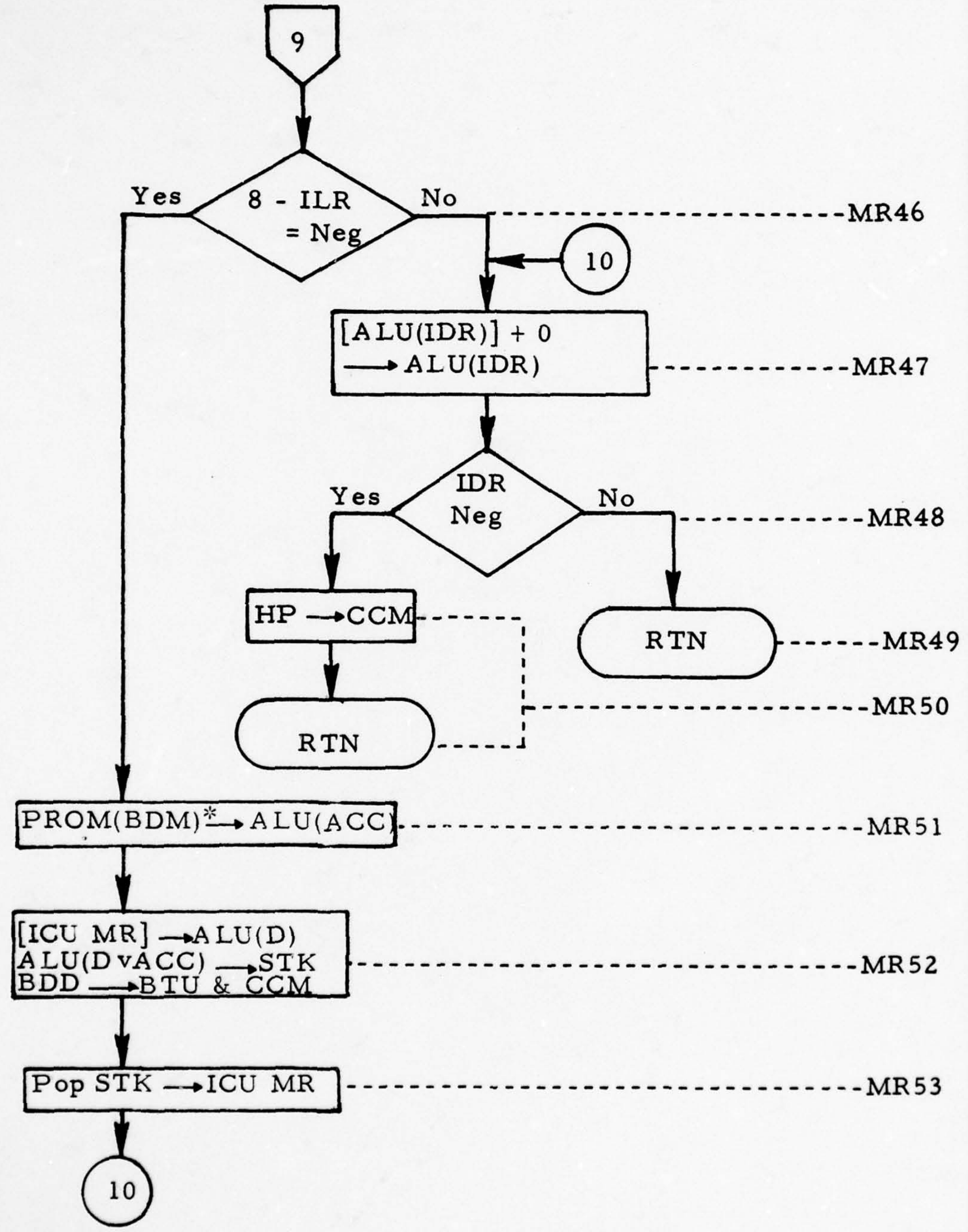
Fig. B-8.B. (continued)



*MBC = Message Buffer Constant, xxxxxx0000000000

*8Con = A 16-bit word constant with a decimal value of eight.

Fig. B-8.B. (continued)



*BDM = Bus Dominance Mask word, a 16-bit word with the BD bit equal to one and the other bits equal to zero.

Fig. B-8.B. (continued)

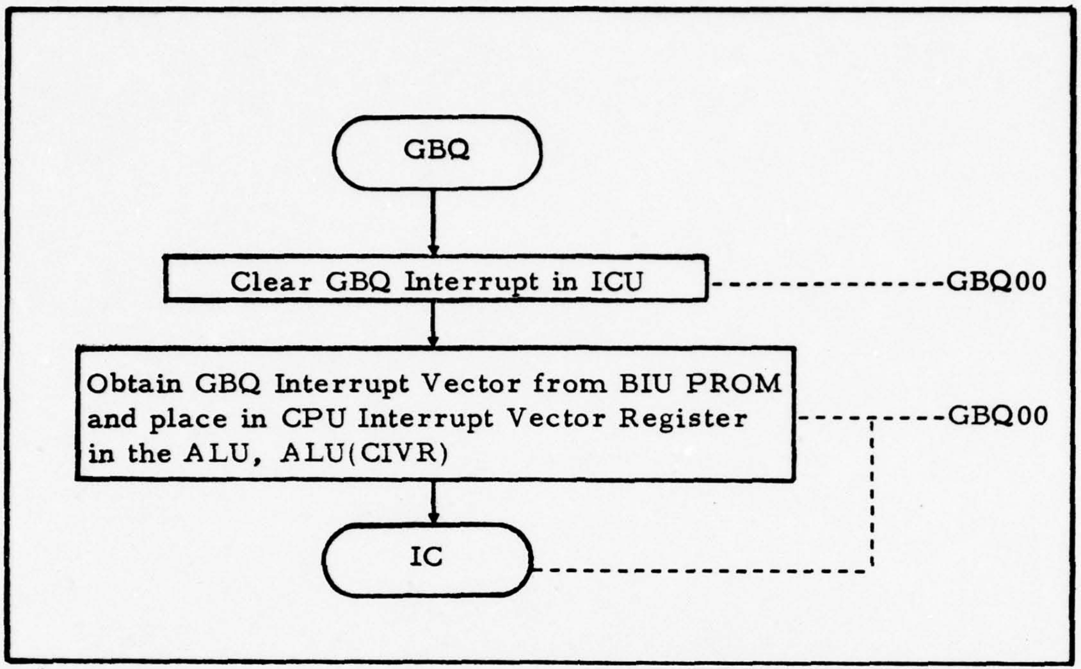

GBQ
Clear GBQ Interrupt in ICU
GBQ00
Obtain GBQ Interrupt Vector from BIU PROM and place in CPU Interrupt Vector Register in the ALU, ALU(CIVR)
GBQ00
IC

Fig. B-9.A. Functional Flowchart for Global Bus Quiescent Routine (GBQ)

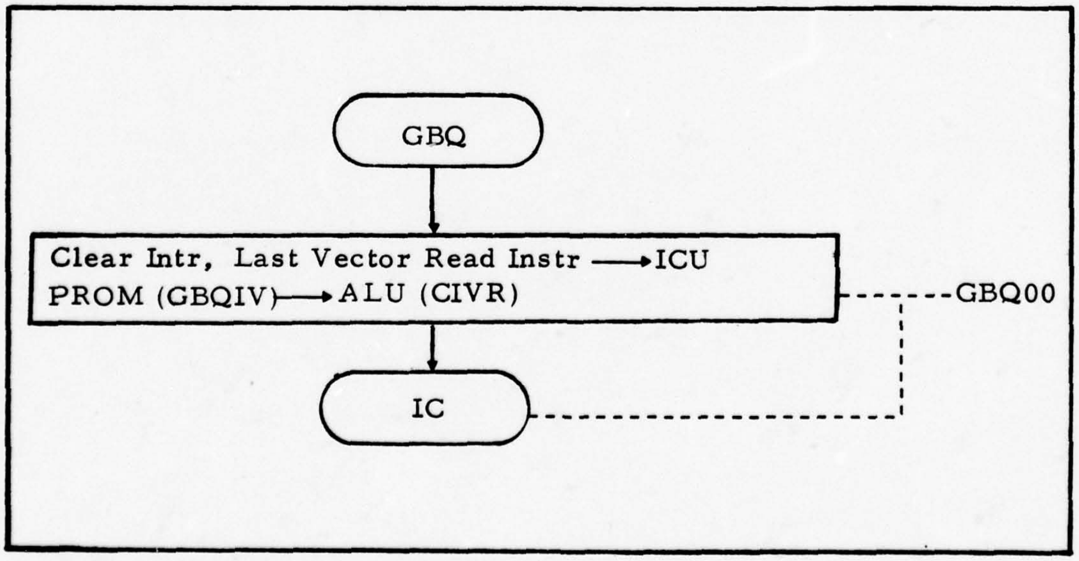

GBQ
Clear Intr, Last Vector Read Instr → ICU
PROM (GBQIV) → ALU (CIVR)
GBQ00
IC

Fig. B-9.B. RTL for Global Bus Quiescent Routine

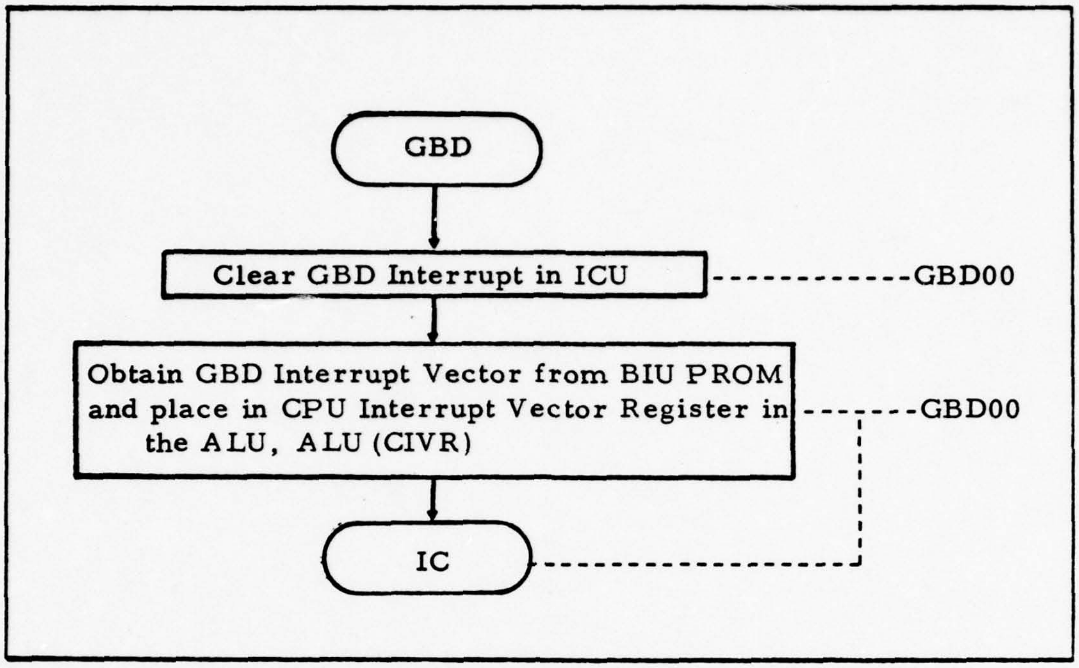

GBD
Clear GBD Interrupt in ICU
GBD00
Obtain GBD Interrupt Vector from BIU PROM and place in CPU Interrupt Vector Register in the ALU, ALU (CIVR)
GBD00
IC

Fig. B-10.A. Functional Flowchart for Global Bus Dominance Routine (GBD)

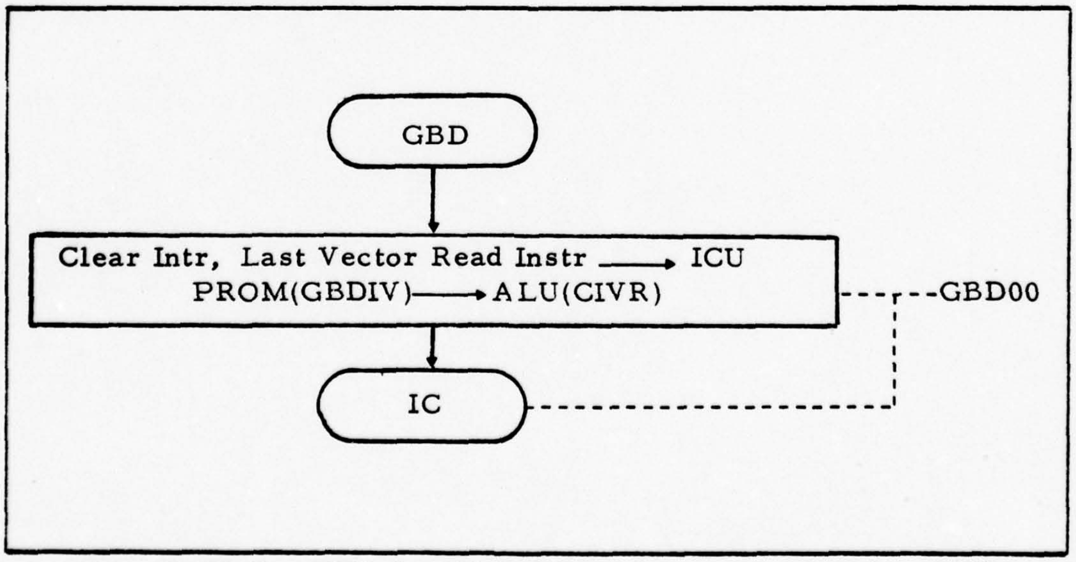


Fig. B-10.B. RTL for Global Bus Dominance Routine (GBD)

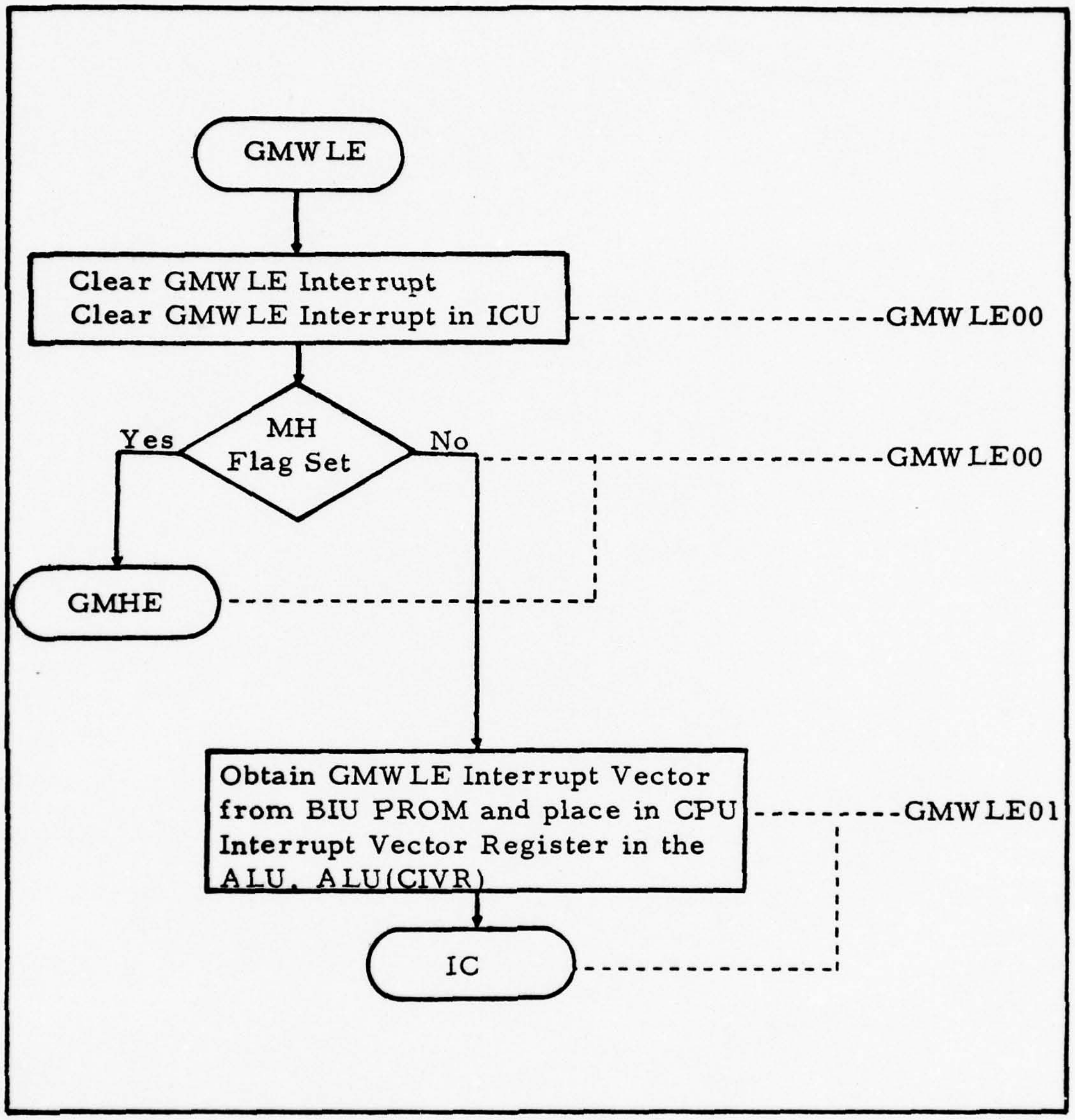

GMWLE
Clear GMWLE Interrupt
Clear GMWLE Interrupt in ICU
GMWLE00
MH Flag Set
Yes
No
GMWLE00
GMHE
Obtain GMWLE Interrupt Vector from BIU PROM and place in CPU Interrupt Vector Register in the ALU. ALU(CIVR)
GMWLE01
IC

Fig. B-11.A. Functional Flowchart for Global Message Word Length Error Routine (GMWLE)

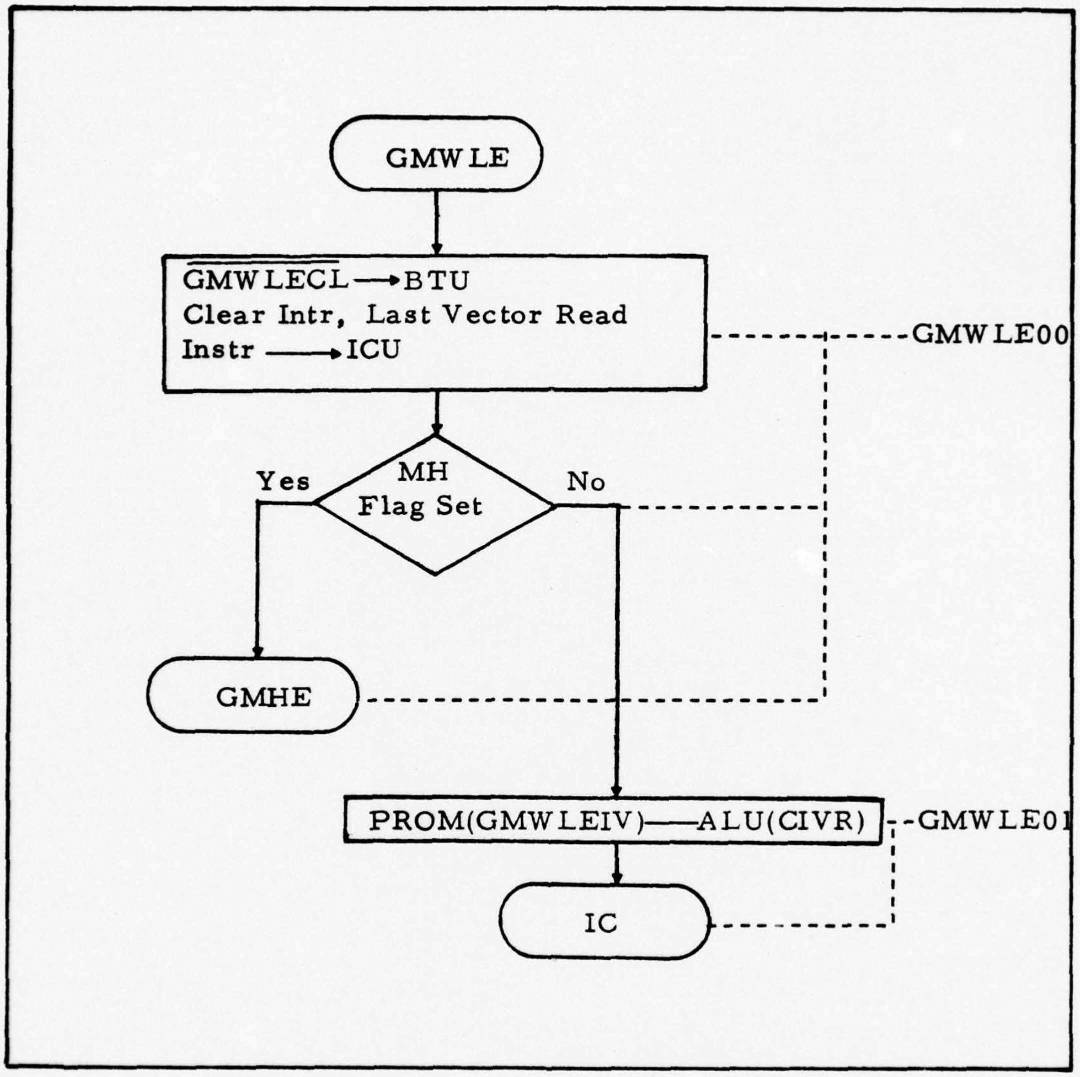


Fig. B-11.B. RTL for Global Message Word Length Error Routine

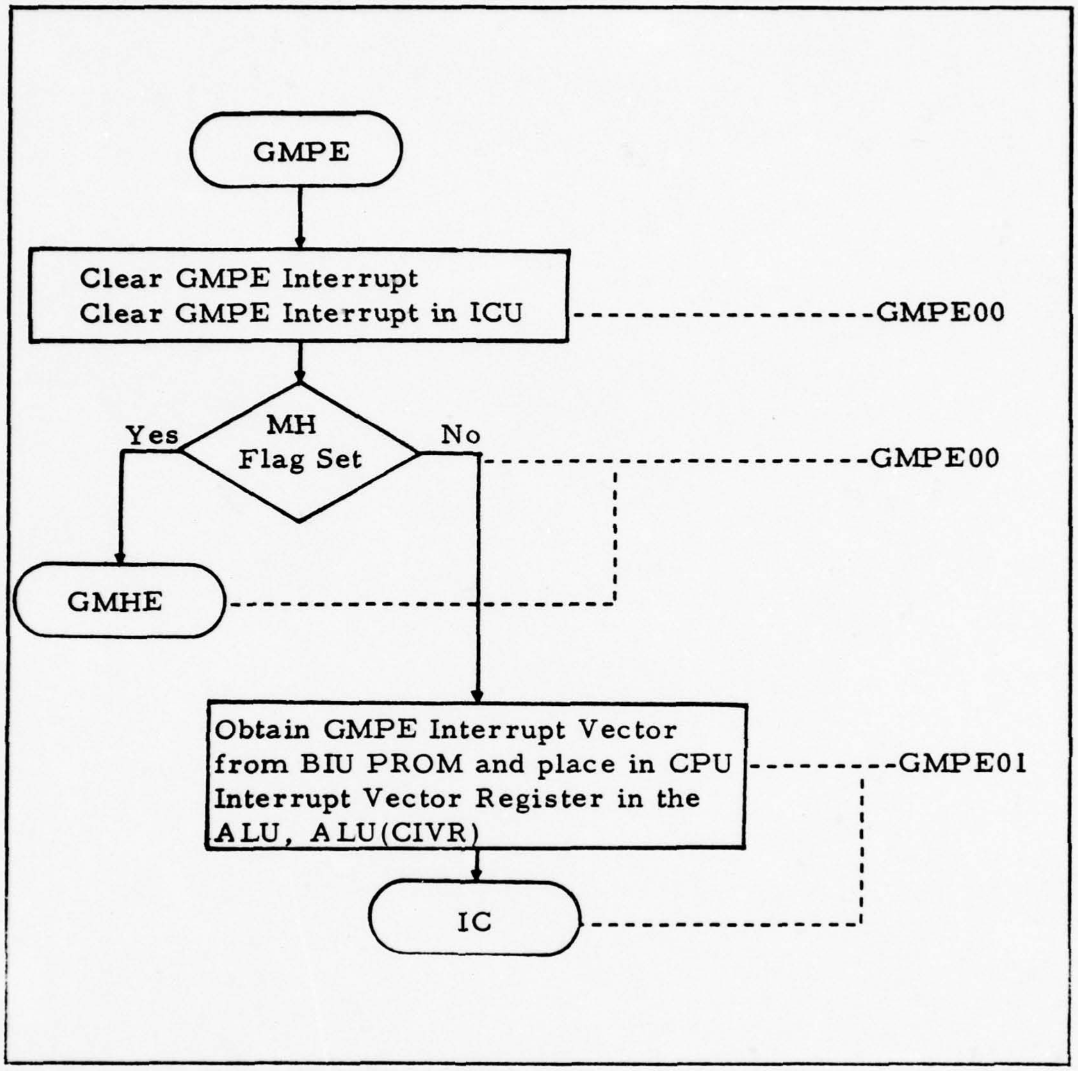

GMPE
Clear GMPE Interrupt
Clear GMPE Interrupt in ICU
GMPE00
MH
Flag Set
Yes
No
GMPE00
GMHE
Obtain GMPE Interrupt Vector from BIU PROM and place in CPU Interrupt Vector Register in the ALU, ALU(CIVR)
GMPE01
IC

Fig. B-12.A. Functional Flowchart for Global Message Parity Error Routine (GMPE)

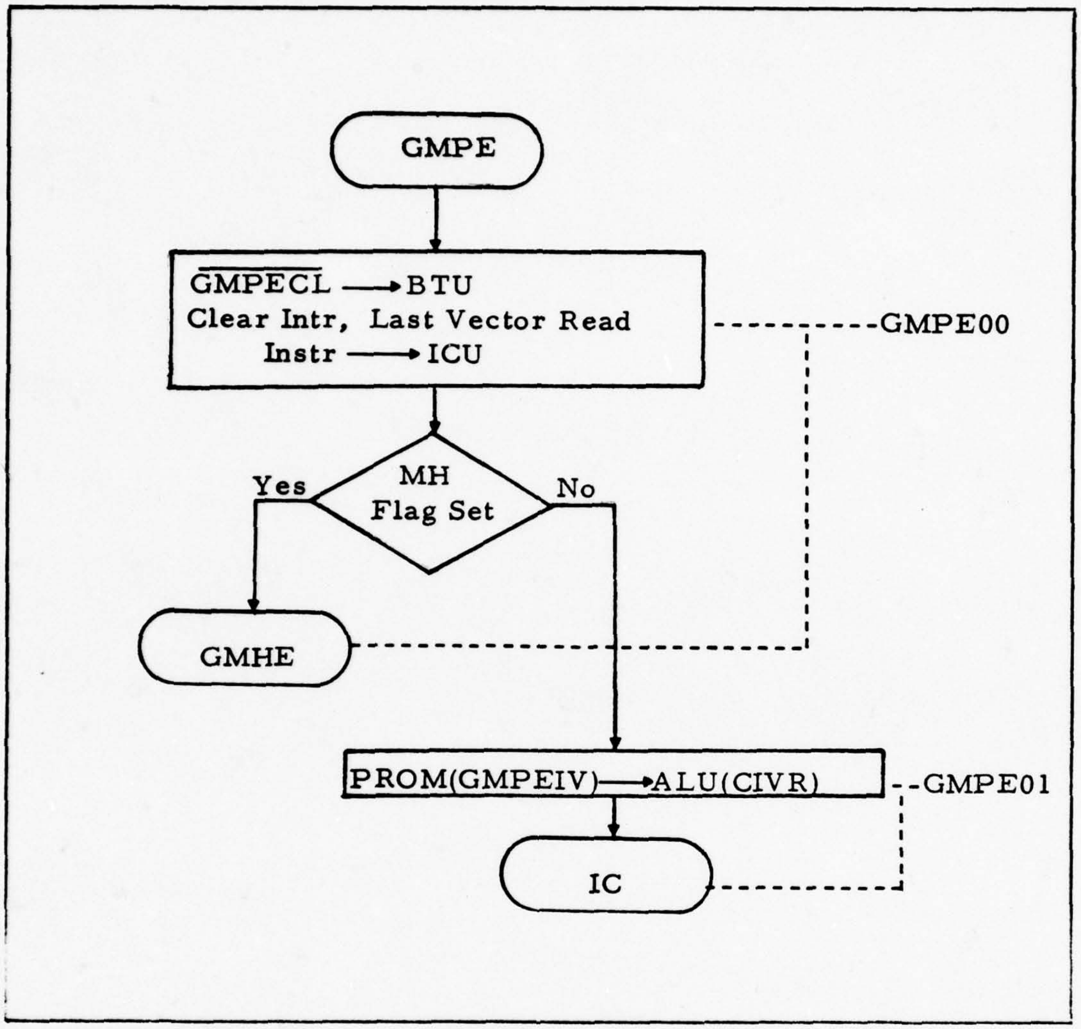


Fig. B-12. B. RTL for Global Message Parity Error Routine (GMPE)

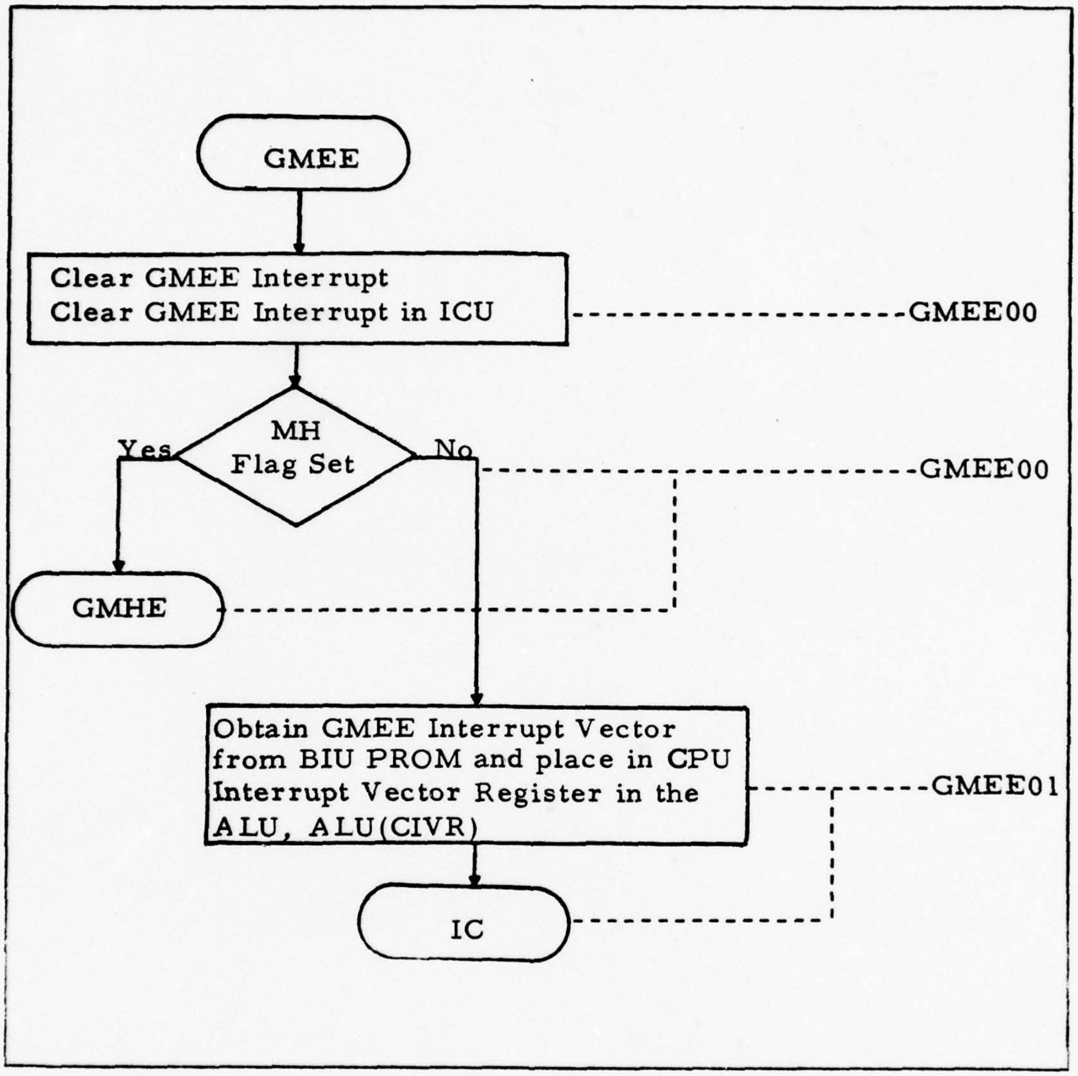

GMEE
Clear GMEE Interrupt
Clear GMEE Interrupt in ICU
GMEE00
MH
Flag Set
Yes
No
GMEE00
GMHE
Obtain GMEE Interrupt Vector from BIU PROM and place in CPU Interrupt Vector Register in the ALU, ALU(CIVR)
GMEE01
IC

Fig. B-13.A. Functional Flowchart for Global Message Encoding Error Routine (GMEE)

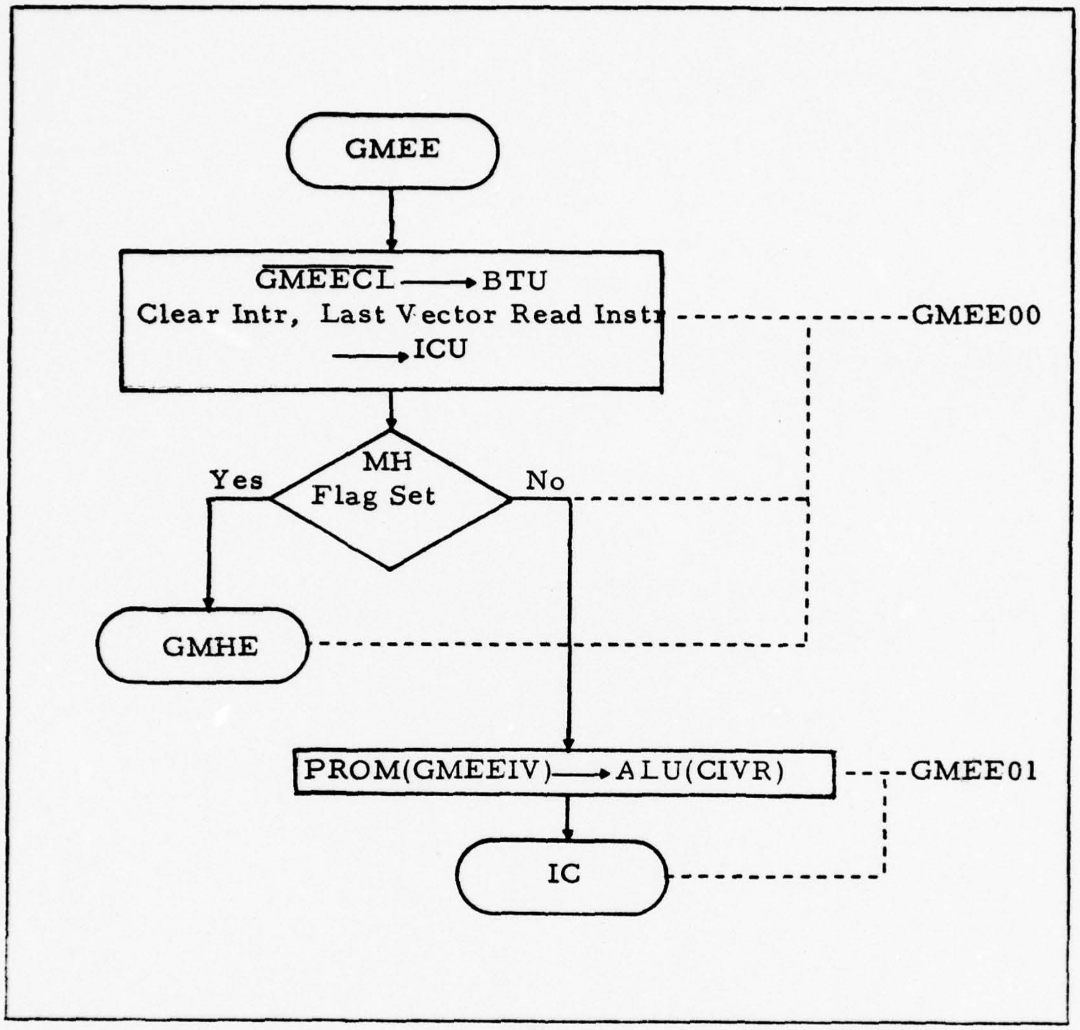


Fig. B-13.B. RTL for Global Message Encoding Error Routine

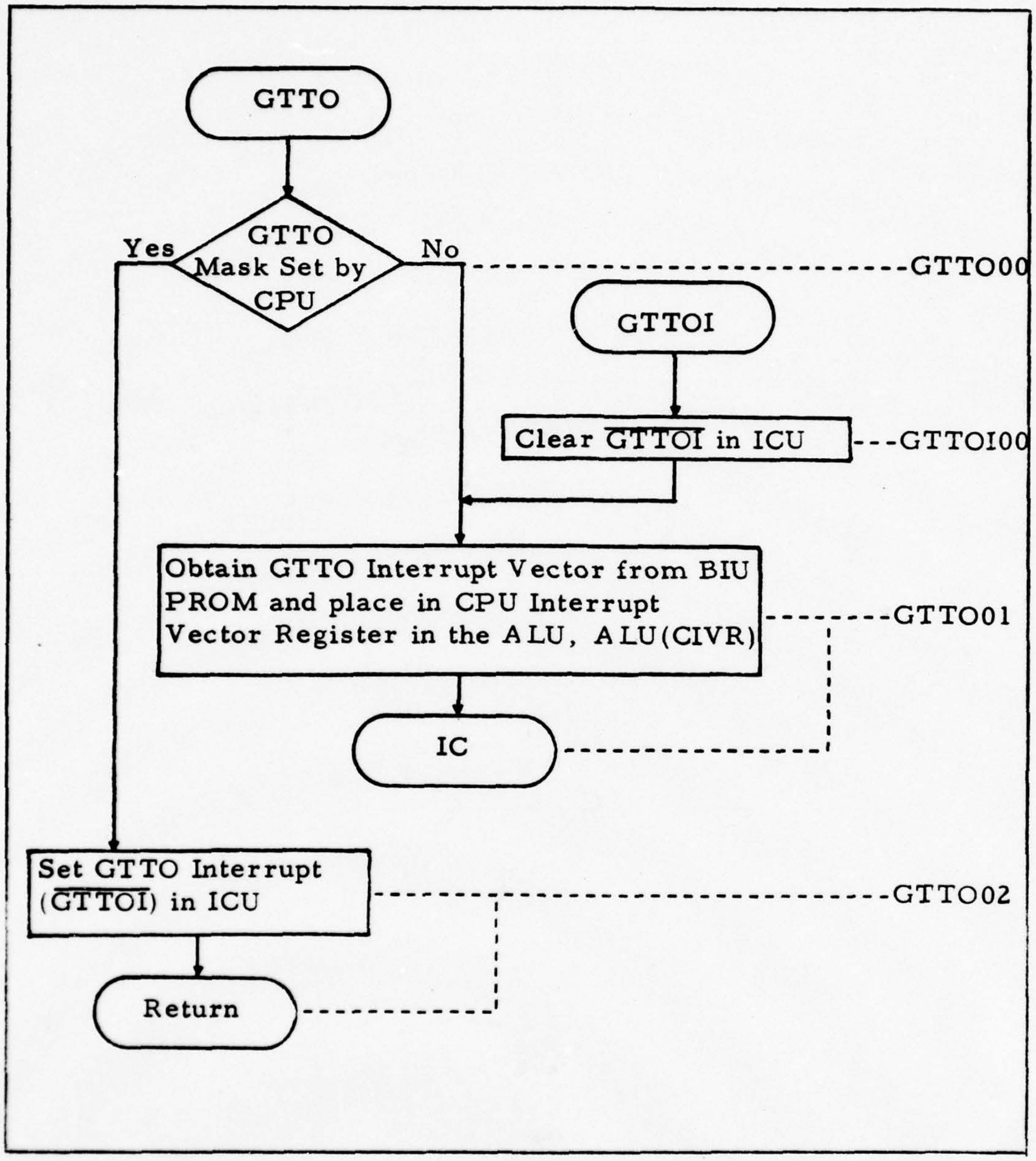

GTTO
GTTO Mask Set by CPU
Yes
No
GTTO00
GTTOI
Clear $\overline{\text{GTTOI}}$ in ICU
GTTOI00
Obtain GTTO Interrupt Vector from BIU PROM and place in CPU Interrupt Vector Register in the ALU, ALU(CIVR)
GTTO01
IC
Set GTTO Interrupt ($\overline{\text{GTTOI}}$) in ICU
GTTO02
Return

Fig. B-14.A. Functional Flowchart for Global Transfer Time Out Routine (GTTO and GTTOI)

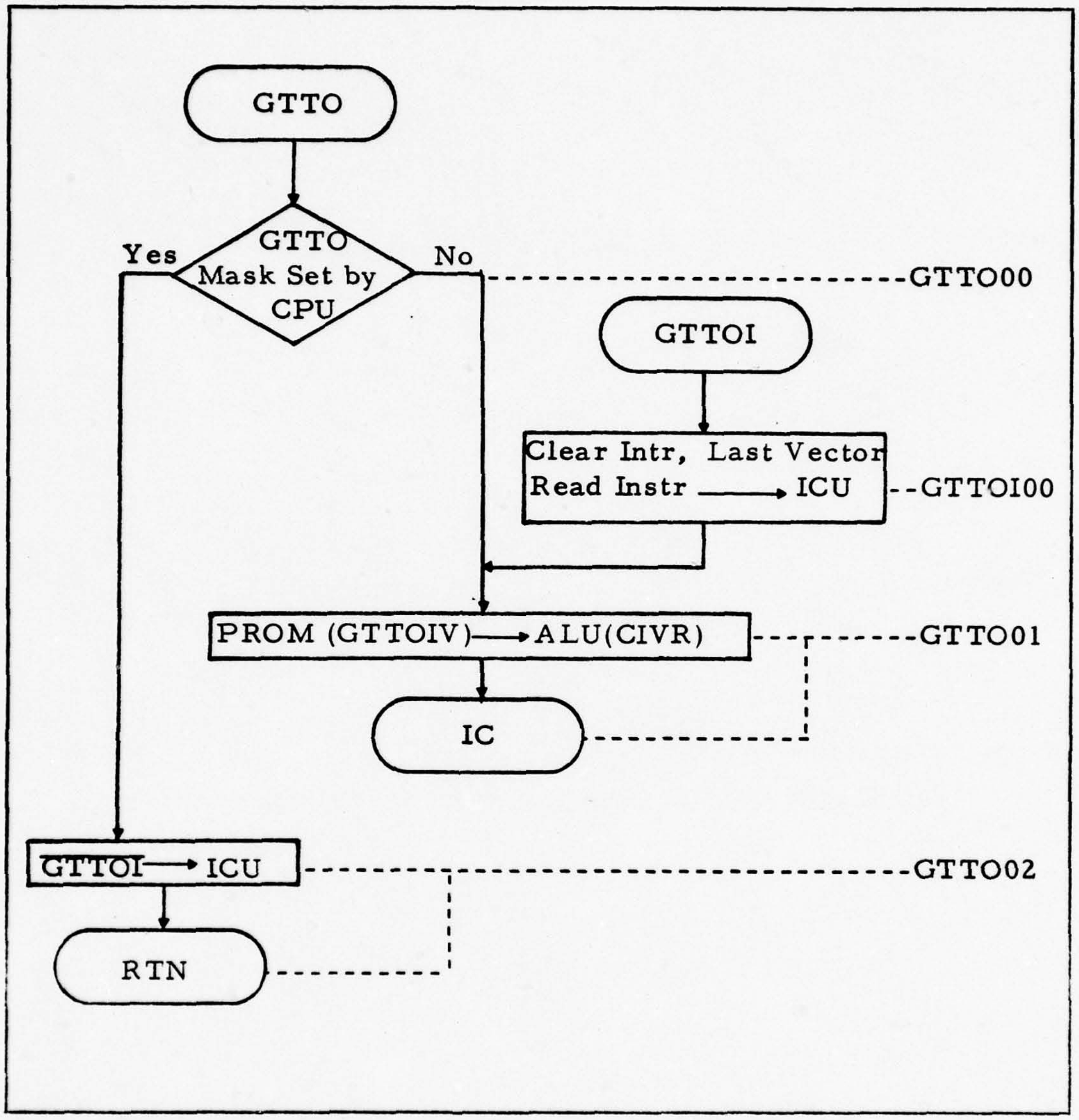


Fig. B-14.B. RTL for Global Transfer Time Out Routine

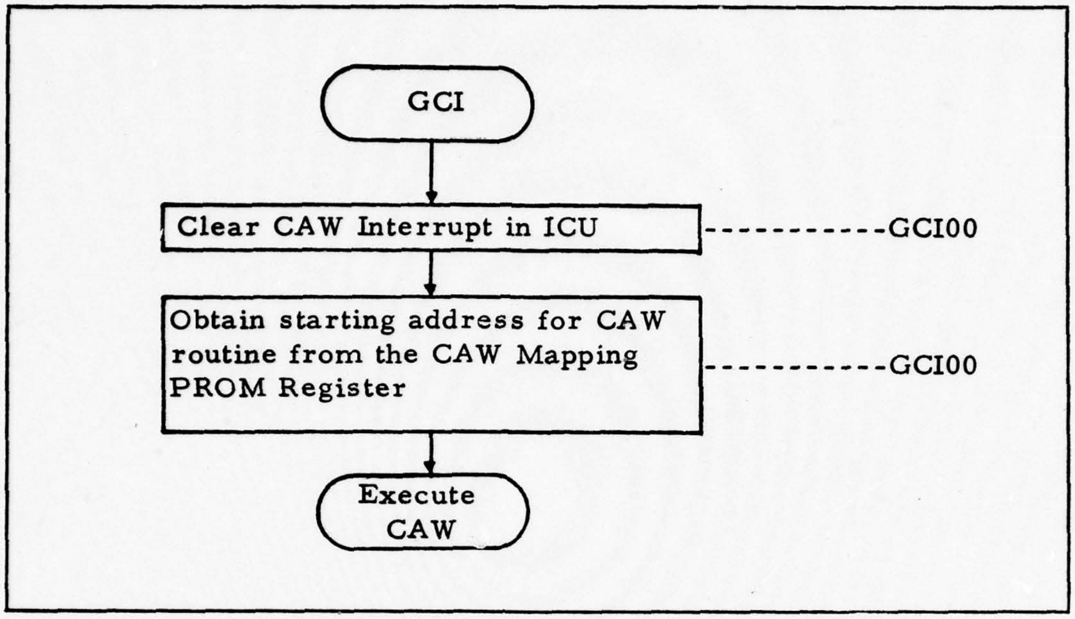

GCI
Clear CAW Interrupt in ICU
GCI00
Obtain starting address for CAW routine from the CAW Mapping PROM Register
GCI00
Execute CAW

Fig. B-15.A. Functional Flowchart for Global CAW Interrupt Routine (GCI)

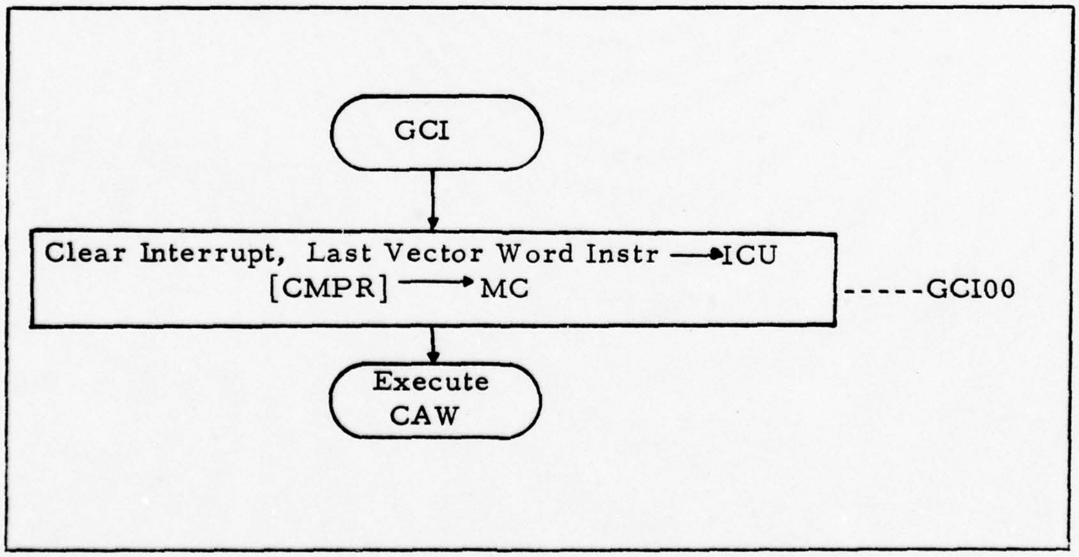


Fig. B-15.B. RTL for Global CAW Interrupt Routine

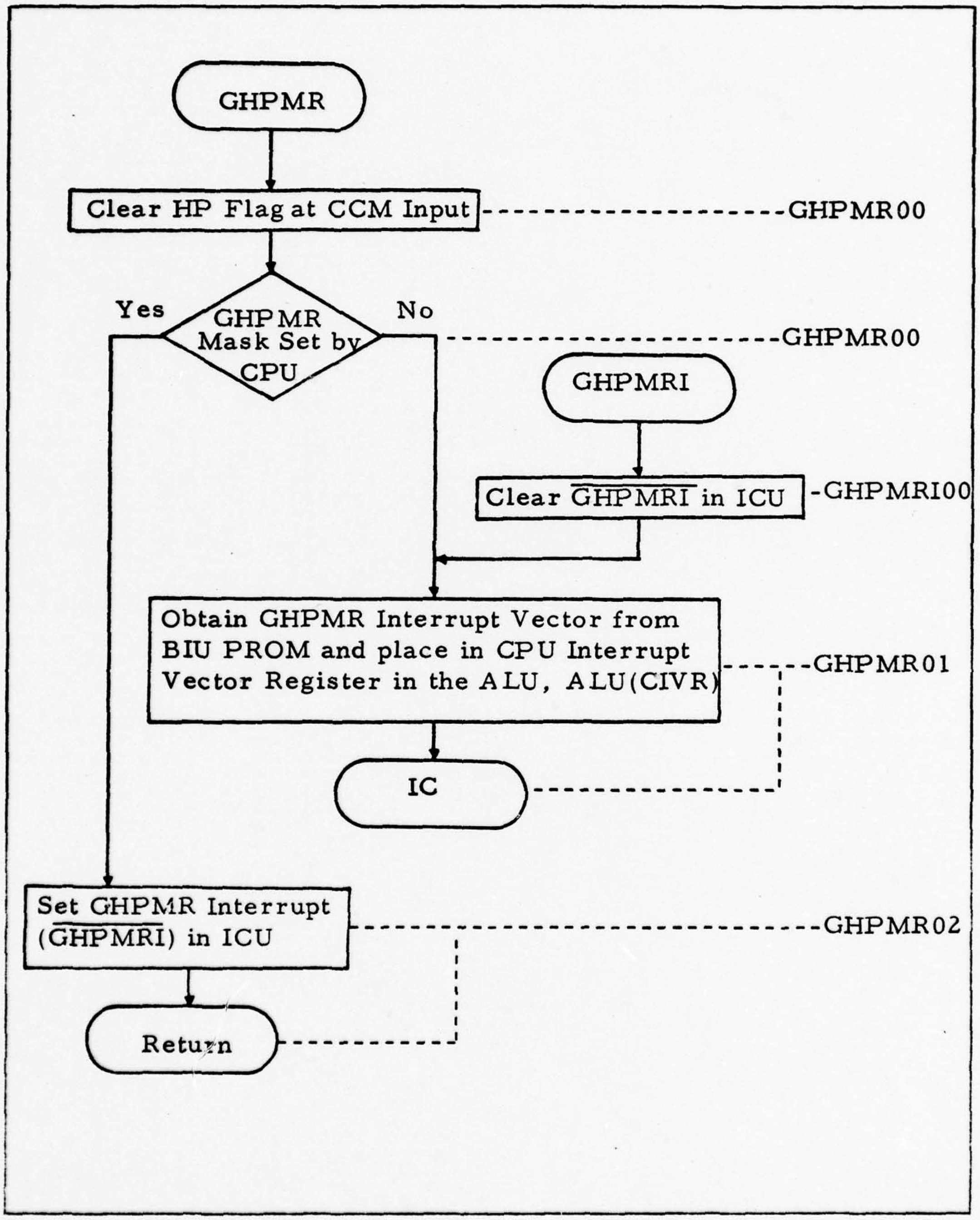

GHPMR
Clear HP Flag at CCM Input
GHPMR00
Yes
GHPMR Mask Set by CPU
No
GHPMR00
GHPMRI
Clear GHPMRI in ICU
GHPMRI00
Obtain GHPMR Interrupt Vector from BIU PROM and place in CPU Interrupt Vector Register in the ALU, ALU(CIVR)
GHPMR01
IC
Set GHPMR Interrupt (GHPMRI) in ICU
GHPMR02
Return

Fig. B-16.A. Functional Flowchart for Global High Priority Message Received Routine (GHPMR and GHPMRI)

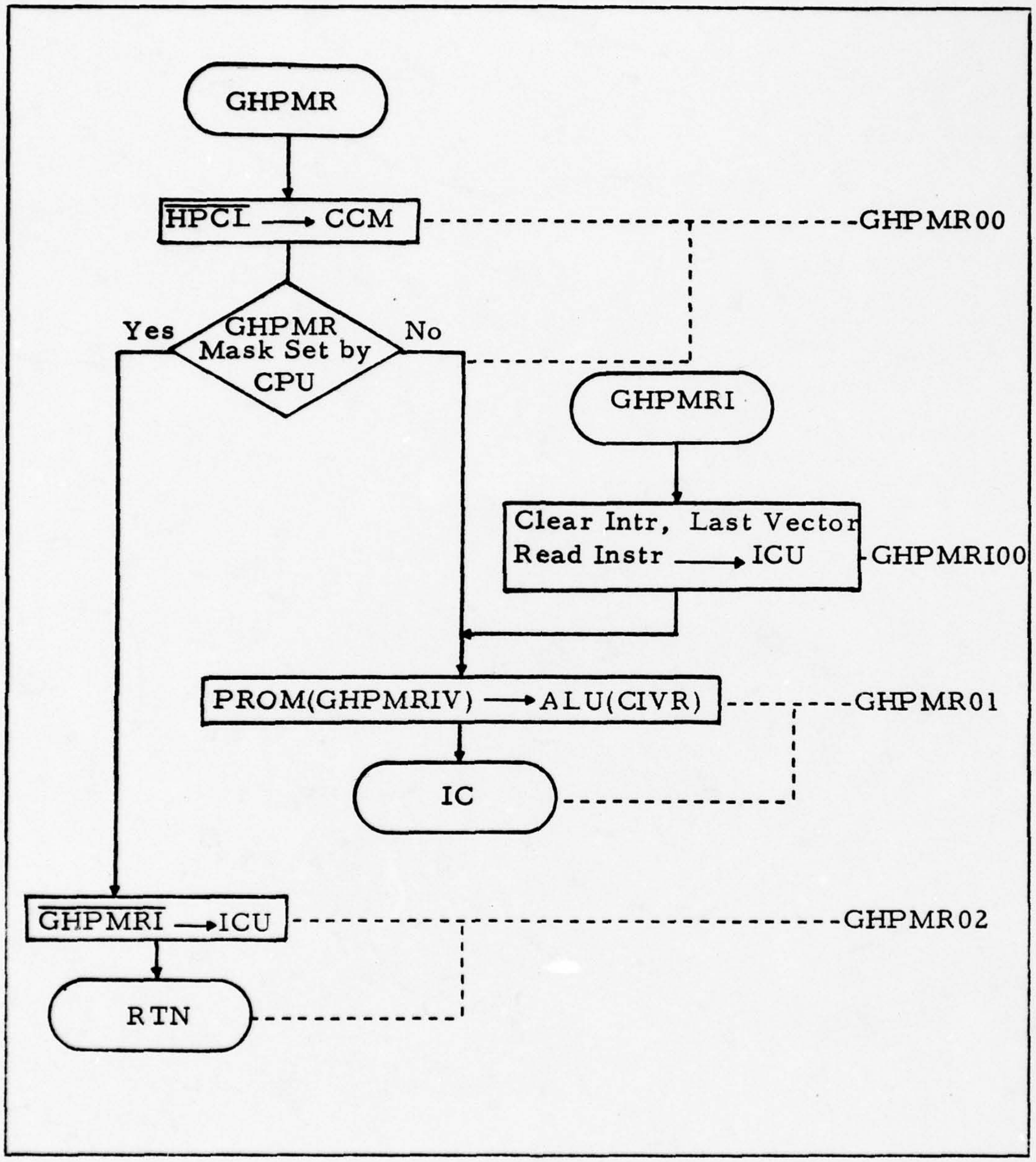


Fig. B-16.B. RTL for Global High Priority Message Received Routine

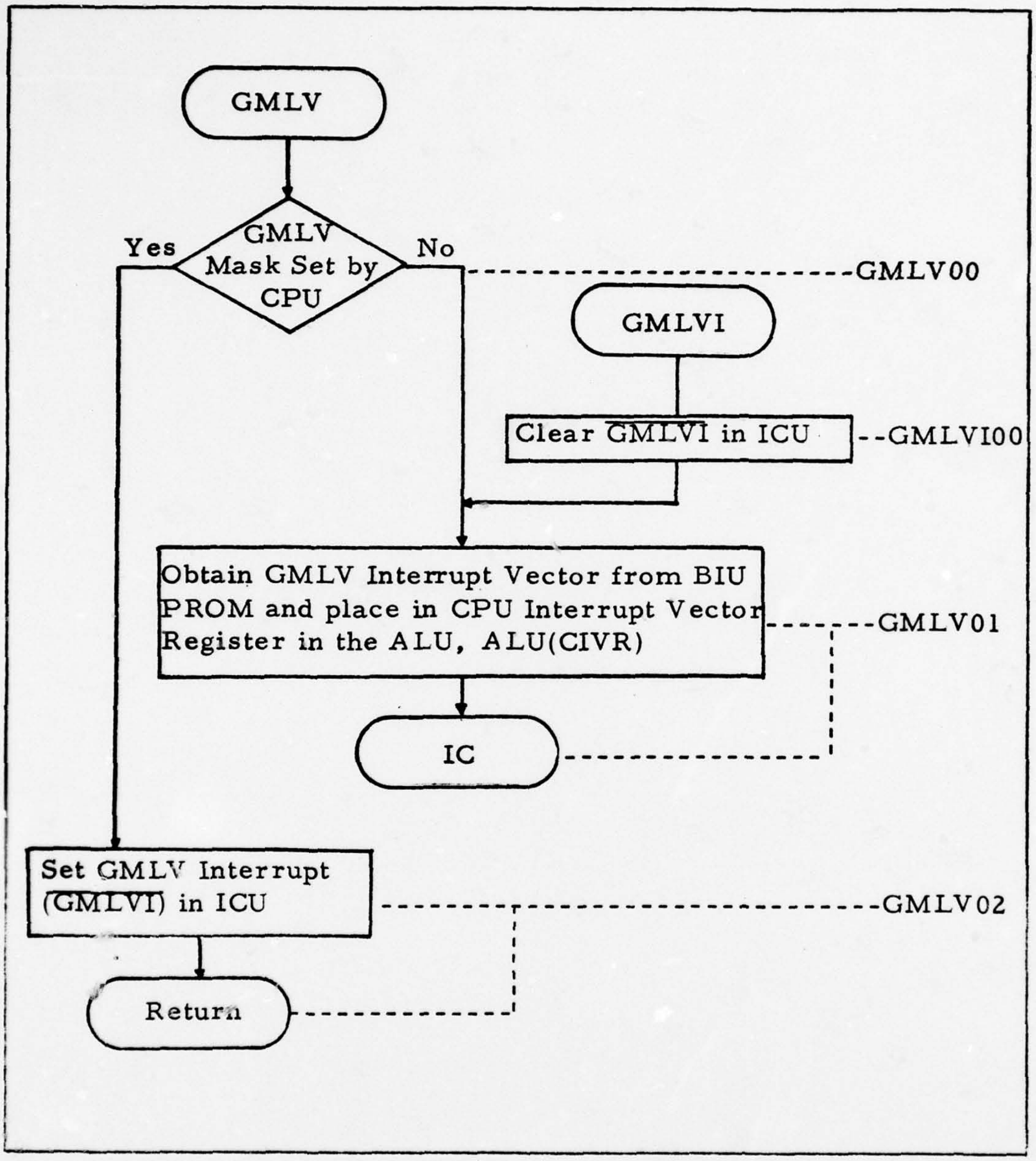

GMLV
GMLV Mask Set by CPU
Yes
No
GMLV00
GMLVI
Clear GMLVI in ICU
GMLVI00
Obtain GMLV Interrupt Vector from BIU PROM and place in CPU Interrupt Vector Register in the ALU, ALU(CIVR)
GMLV01
IC
Set GMLV Interrupt (GMLVI) in ICU
GMLV02
Return

Fig. B-17.A. Functional Flowchart for Global Message Length Violation Routine (GMLV and GMLVI)

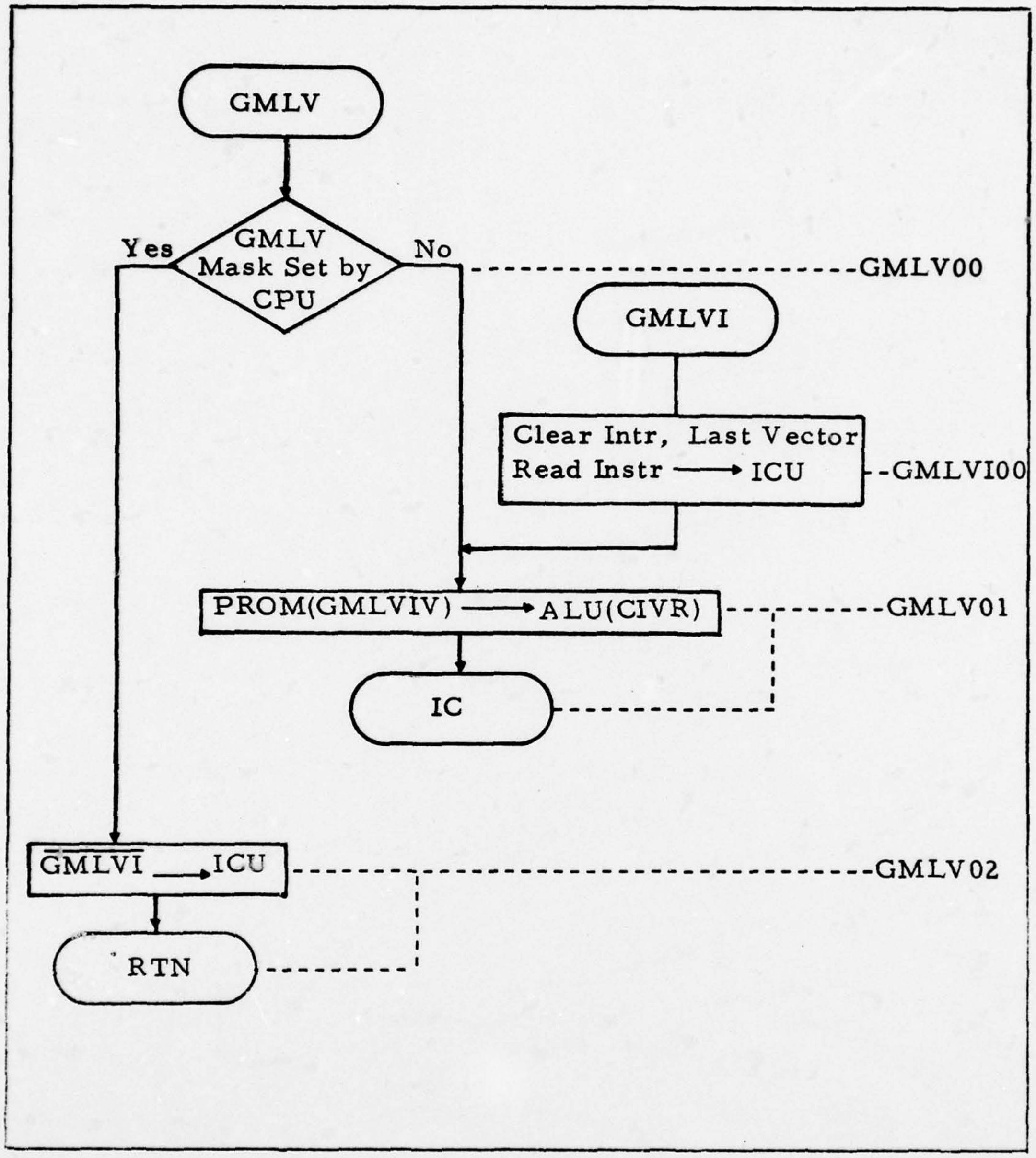
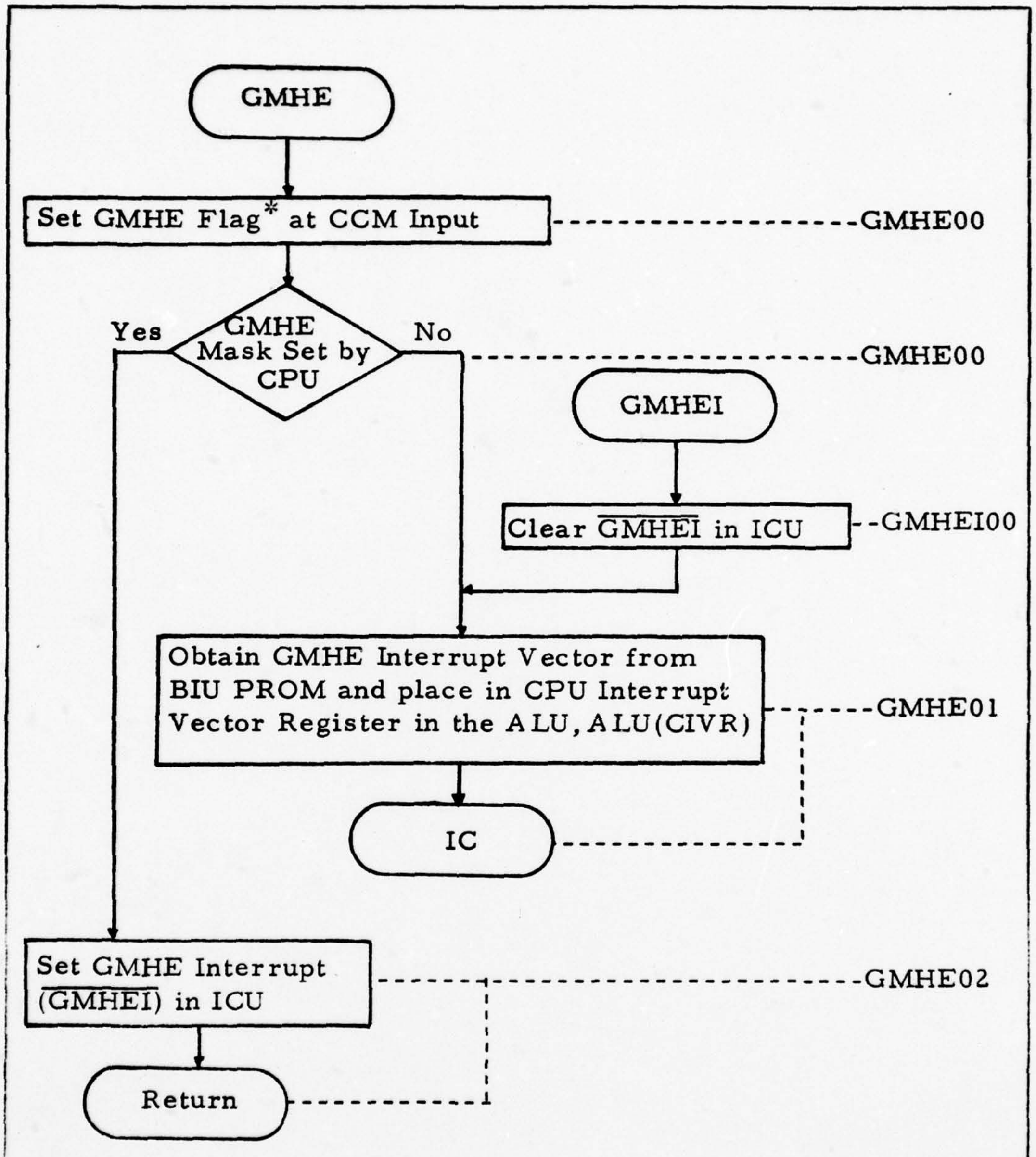


Fig. B-17.B. RTL for Global Message Length Violation Routine


GMHE
Set GMHE Flag* at CCM Input
GMHE00
Yes
GMHE Mask Set by CPU
No
GMHE00
GMHEI
Clear GMHEI in ICU
GMHEI00
Obtain GMHE Interrupt Vector from BIU PROM and place in CPU Interrupt Vector Register in the ALU, ALU(CIVR)
GMHE01
IC
Set GMHE Interrupt (GMHEI) in ICU
GMHE02
Return

*GMHE Flag used in Message Reception Routine (MR) to stop message processing. GMHE Flag is cleared with MHS of next message.

Fig. B-18.A. Functional Flowchart for Global Message Header Error Routine (GMHE and GMHEI)

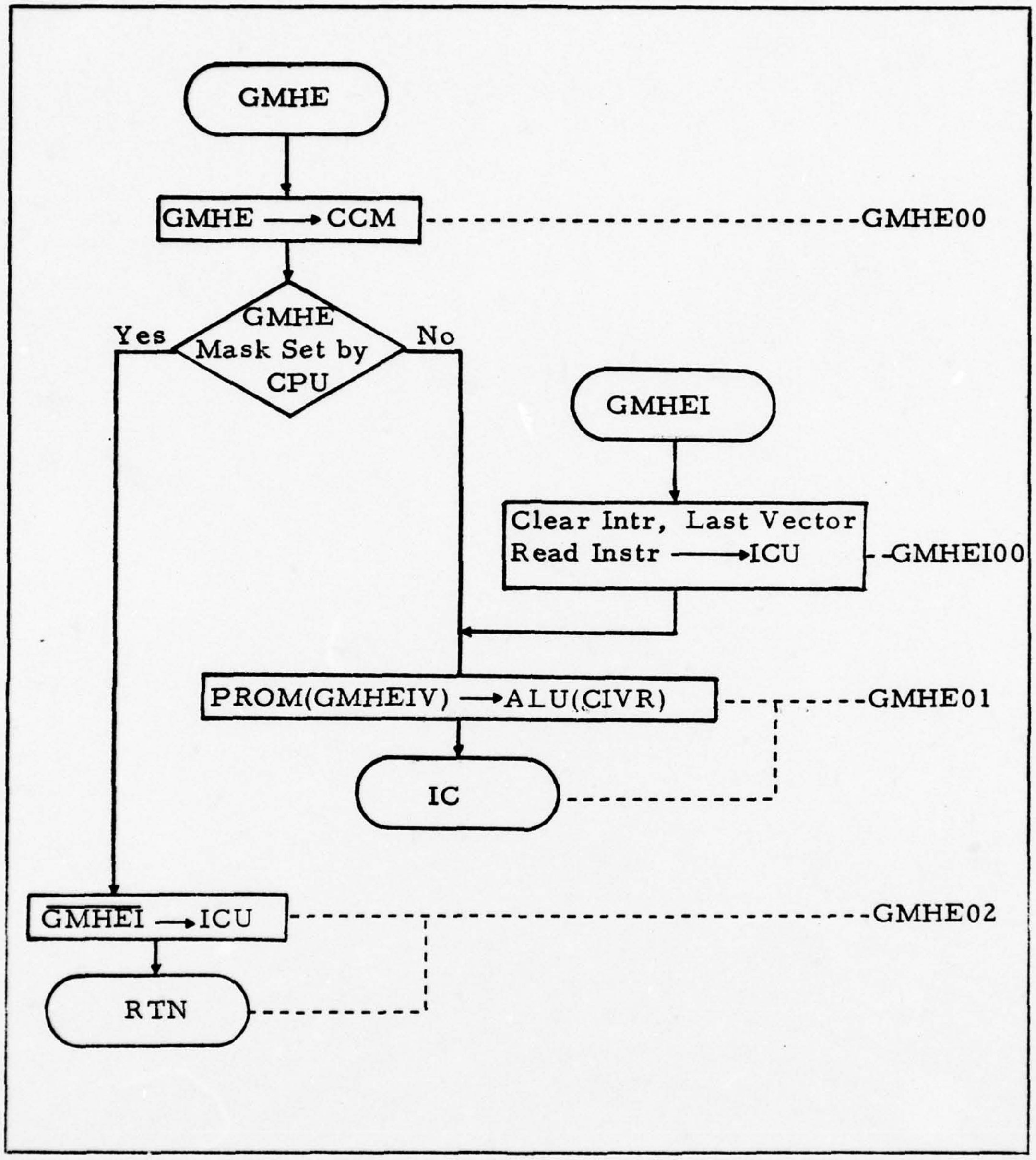


Fig. B-18.B. RTL for Global Message Header Error Routine

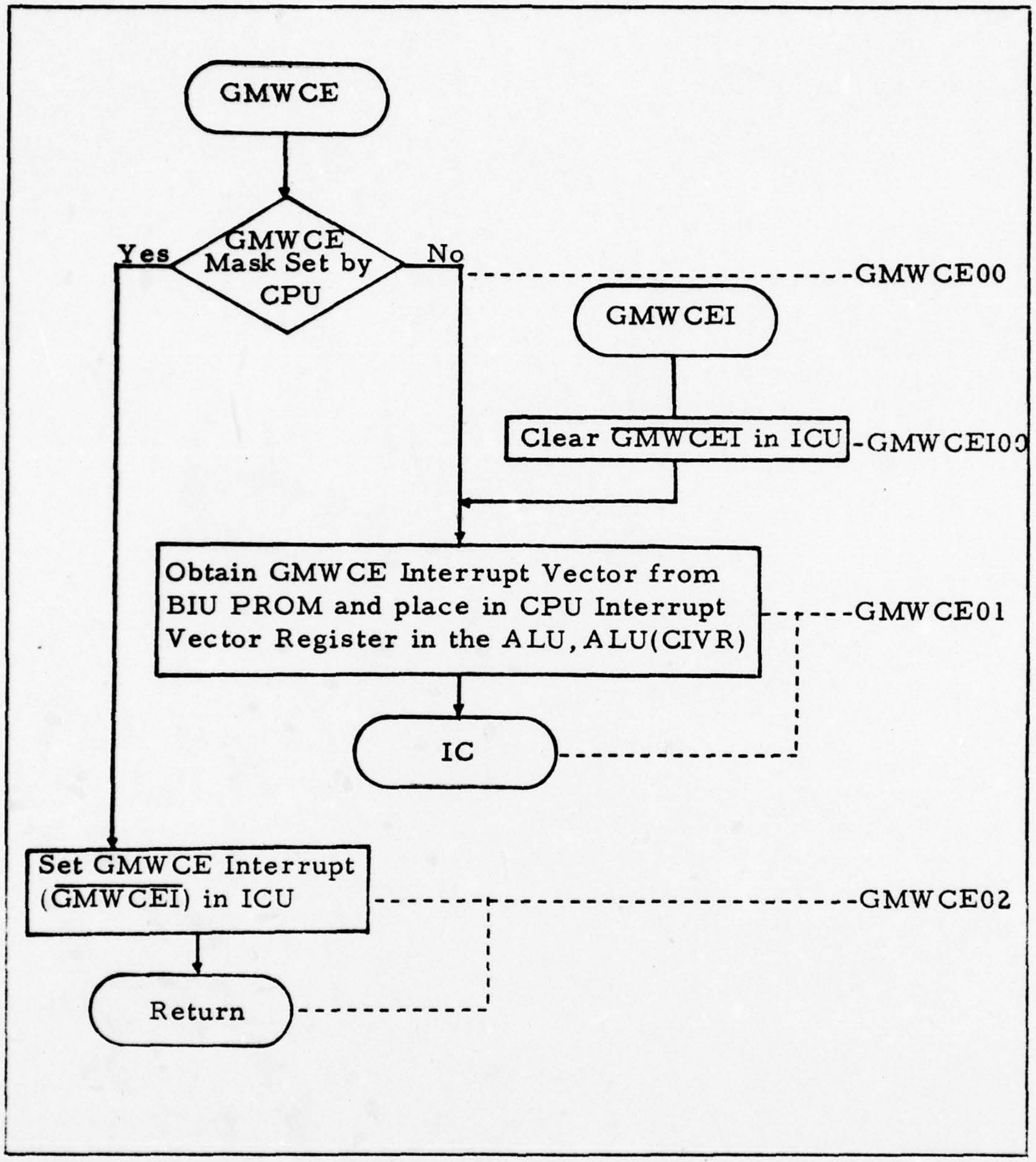

GMWCE
GMWCE Mask Set by CPU
Yes
No
GMWCE00
GMWCEI
Clear GMWCEI in ICU
GMWCEI00
Obtain GMWCE Interrupt Vector from BIU PROM and place in CPU Interrupt Vector Register in the ALU, ALU(CIVR)
GMWCE01
IC
Set GMWCE Interrupt (GMWCEI) in ICU
GMWCE02
Return

Fig. B-19.A. Functional Flowchart for Global Message Word Count Error Routine (GMWCE and GMWCEI)

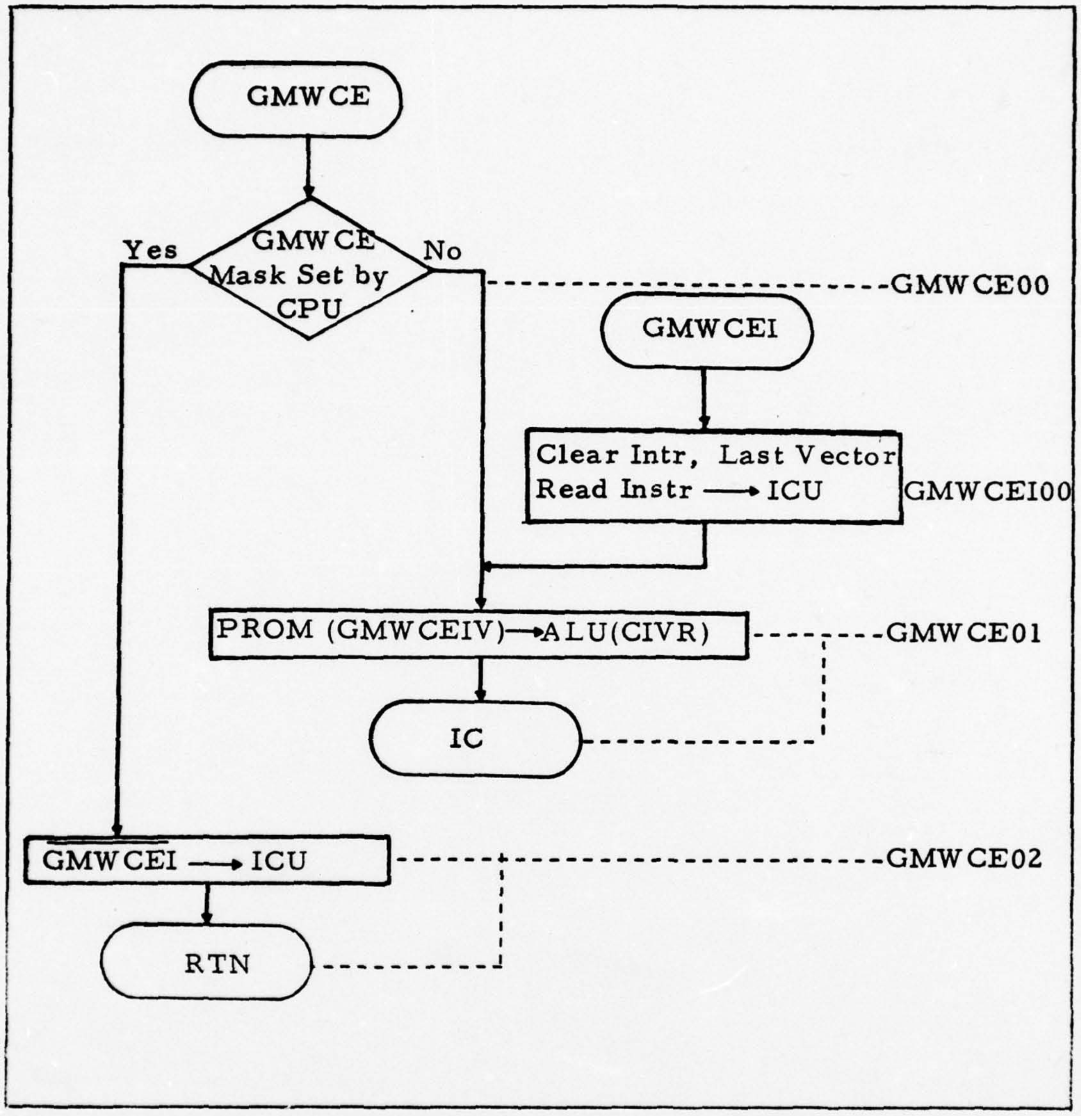


Fig. B-19.B. RTL for Global Message Word Count Error Routine

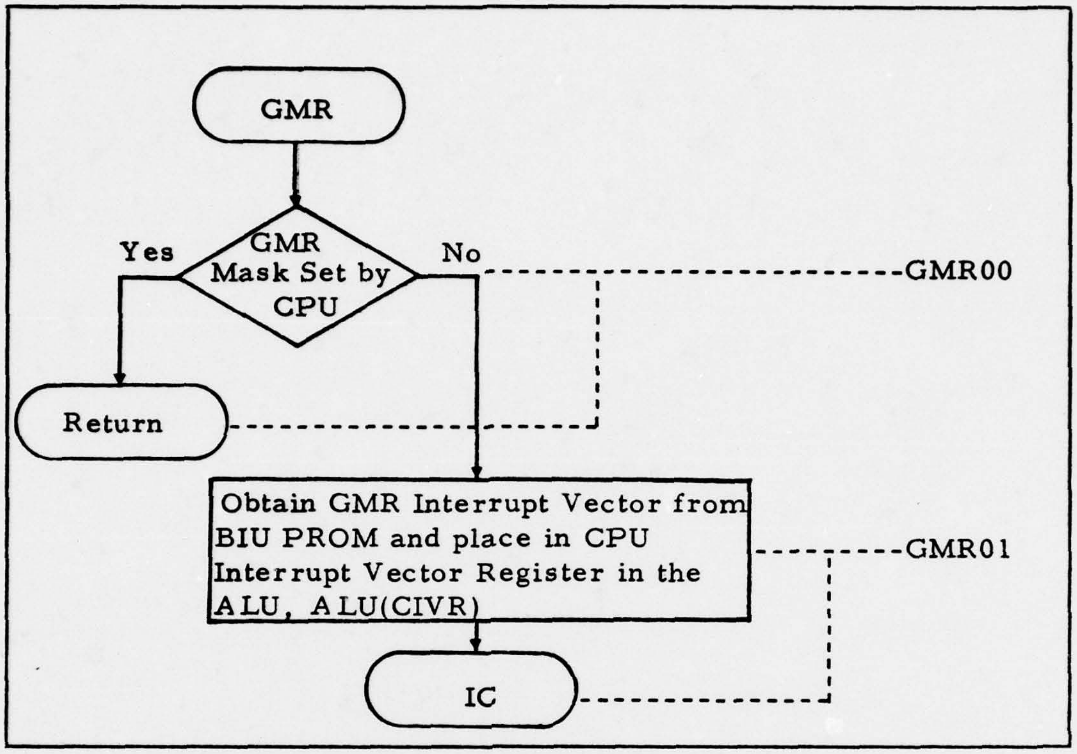

GMR
GMR Mask Set by CPU
Yes
No
Return
Obtain GMR Interrupt Vector from BIU PROM and place in CPU Interrupt Vector Register in the ALU, ALU(CIVR)
IC
GMR00
GMR01

Fig. B-20.A. Functional Flowchart for Global Message Received Routine (GMR)

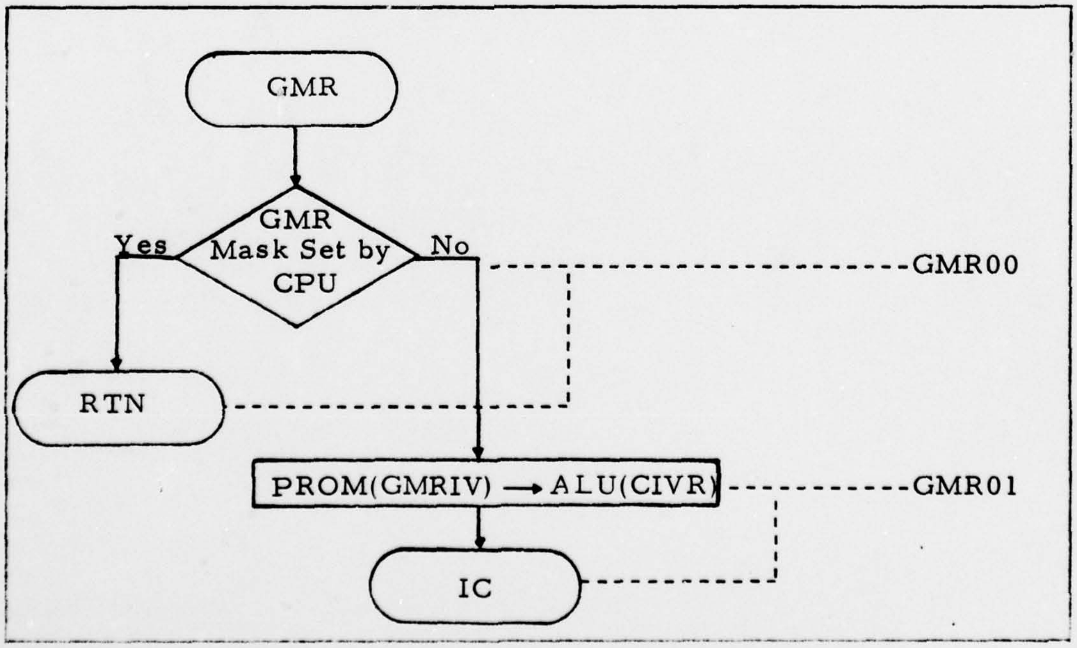


Fig. B-20.B. RTL for Global Message Received Routine

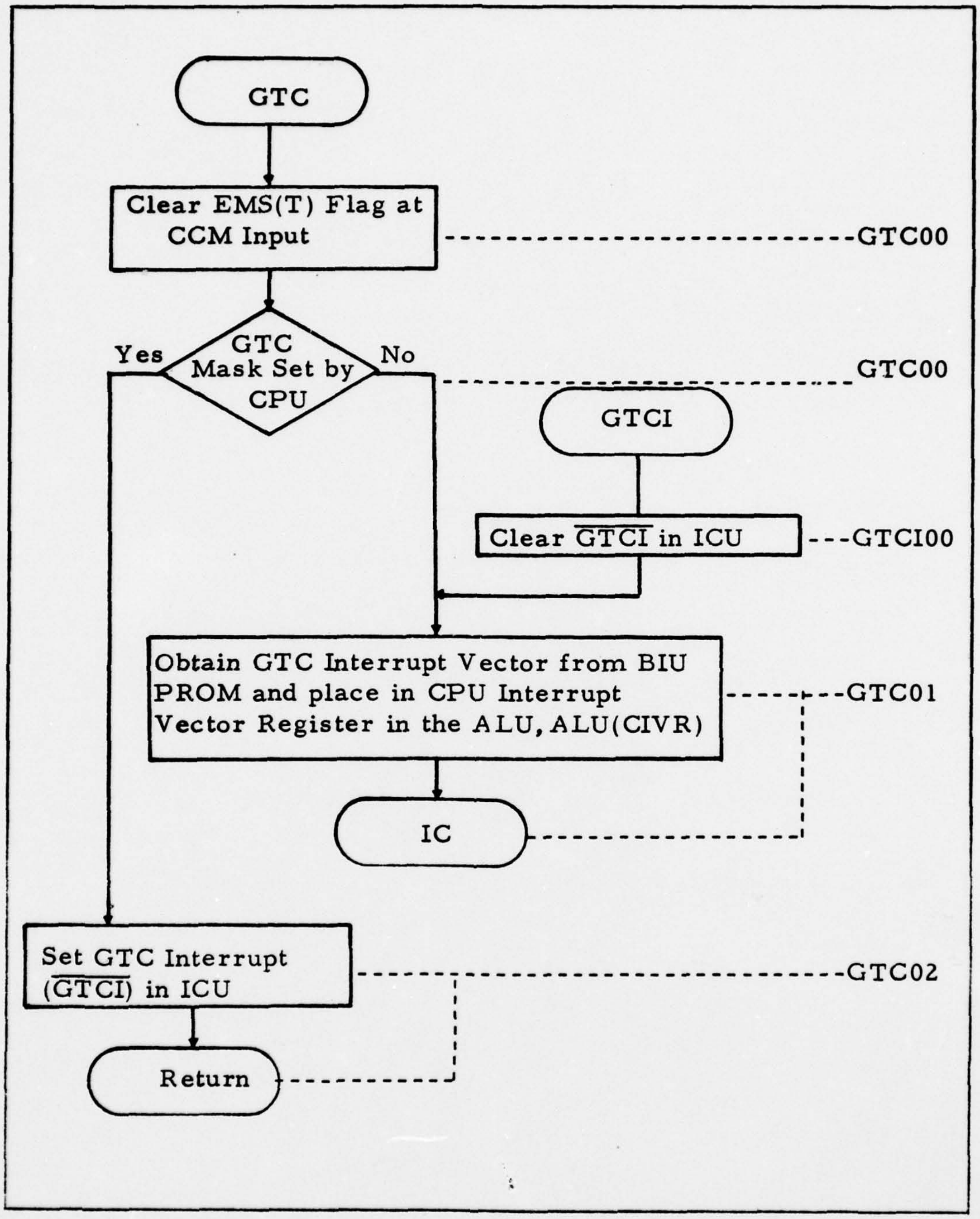

GTC
Clear EMS(T) Flag at CCM Input
GTC00
Yes
GTC Mask Set by CPU
No
GTC00
GTCI
Clear GTCI in ICU
GTCI00
Obtain GTC Interrupt Vector from BIU PROM and place in CPU Interrupt Vector Register in the ALU, ALU(CIVR)
GTC01
IC
Set GTC Interrupt (GTCI) in ICU
GTC02
Return

Fig. B-21.A. Functional Flowchart for Global Transmission Completed Routine (GTC and GTCI)

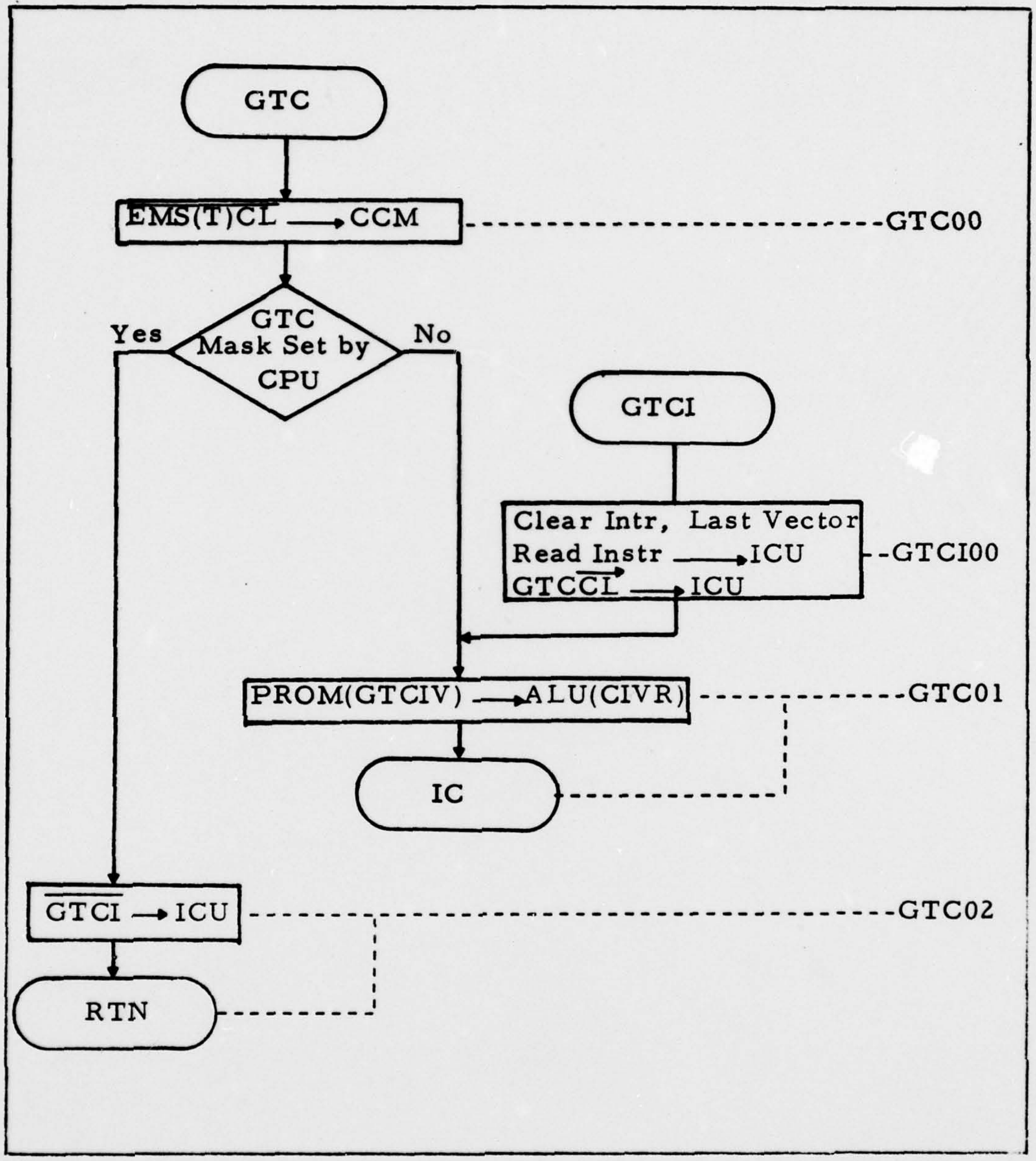


Fig. B-21.B. RTL for Global Transmission Completed Routine

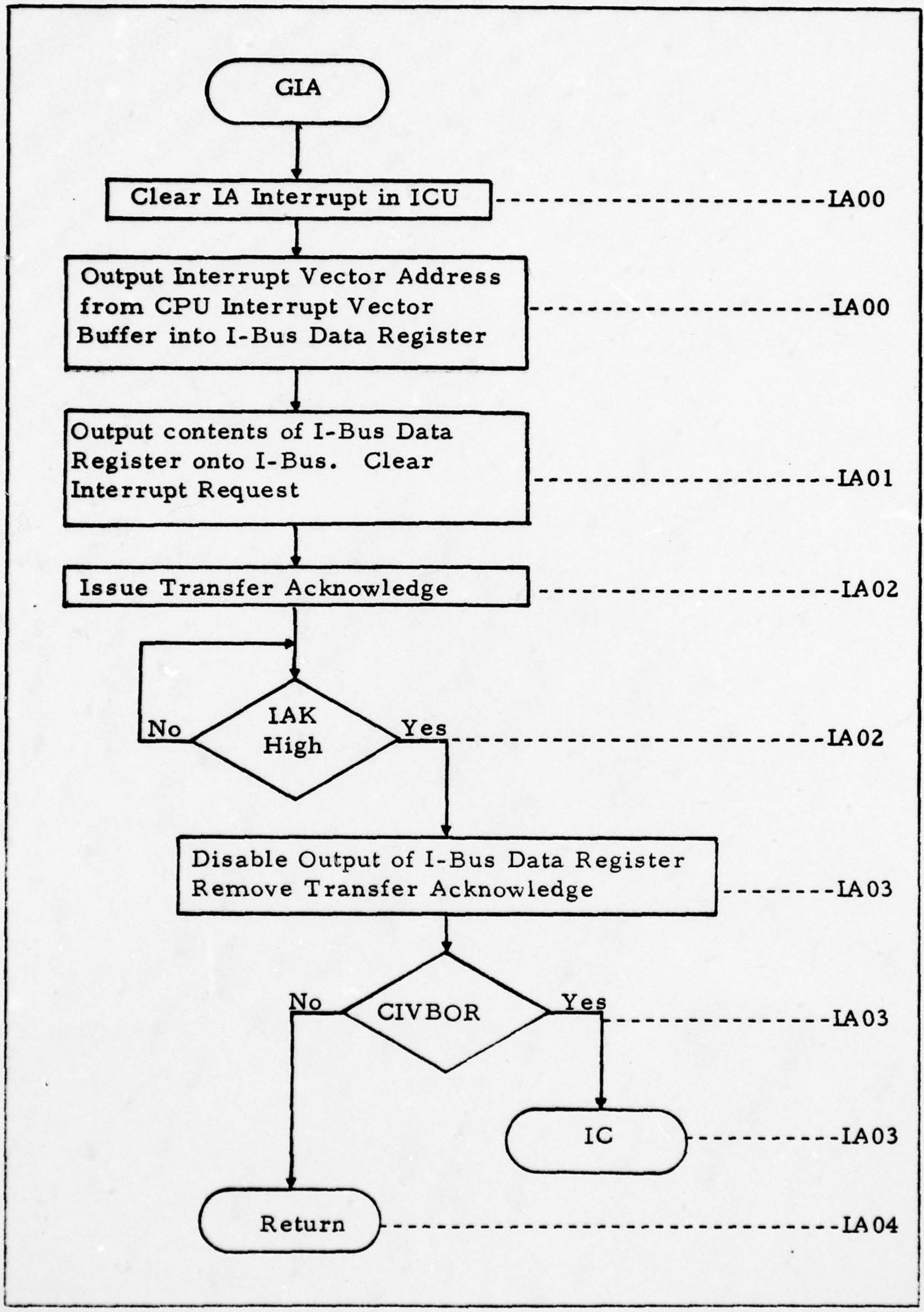

GIA
Clear IA Interrupt in ICU
IA00
Output Interrupt Vector Address from CPU Interrupt Vector Buffer into I-Bus Data Register
IA00
Output contents of I-Bus Data Register onto I-Bus. Clear Interrupt Request
IA01
Issue Transfer Acknowledge
IA02
IAK High
No
Yes
IA02
Disable Output of I-Bus Data Register Remove Transfer Acknowledge
IA03
CIVBOR
No
Yes
IA03
IC
IA03
Return
IA04

Fig. B-22.A. Functional Flowchart for Global Interrupt Acknowledge Routine (GIA)

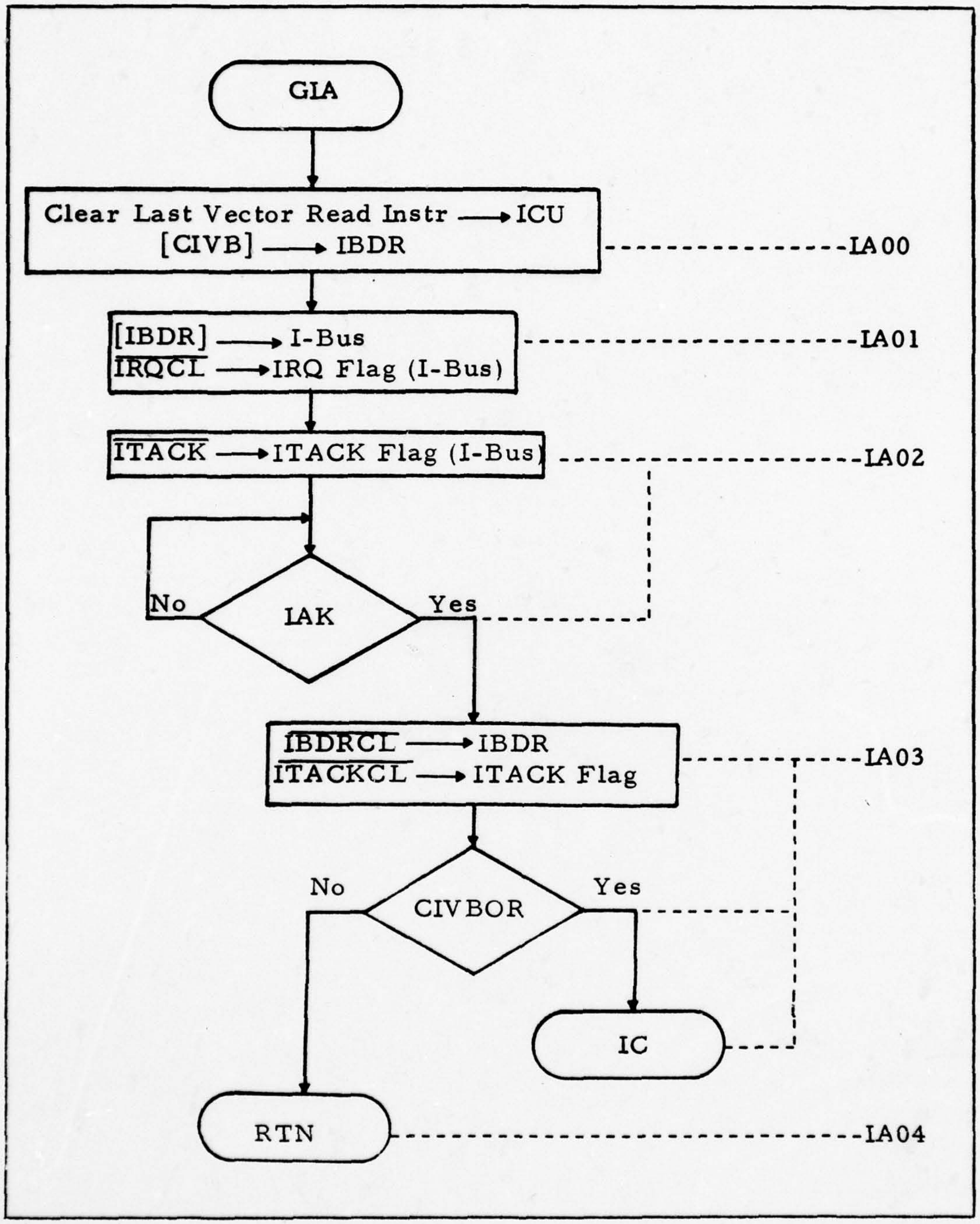
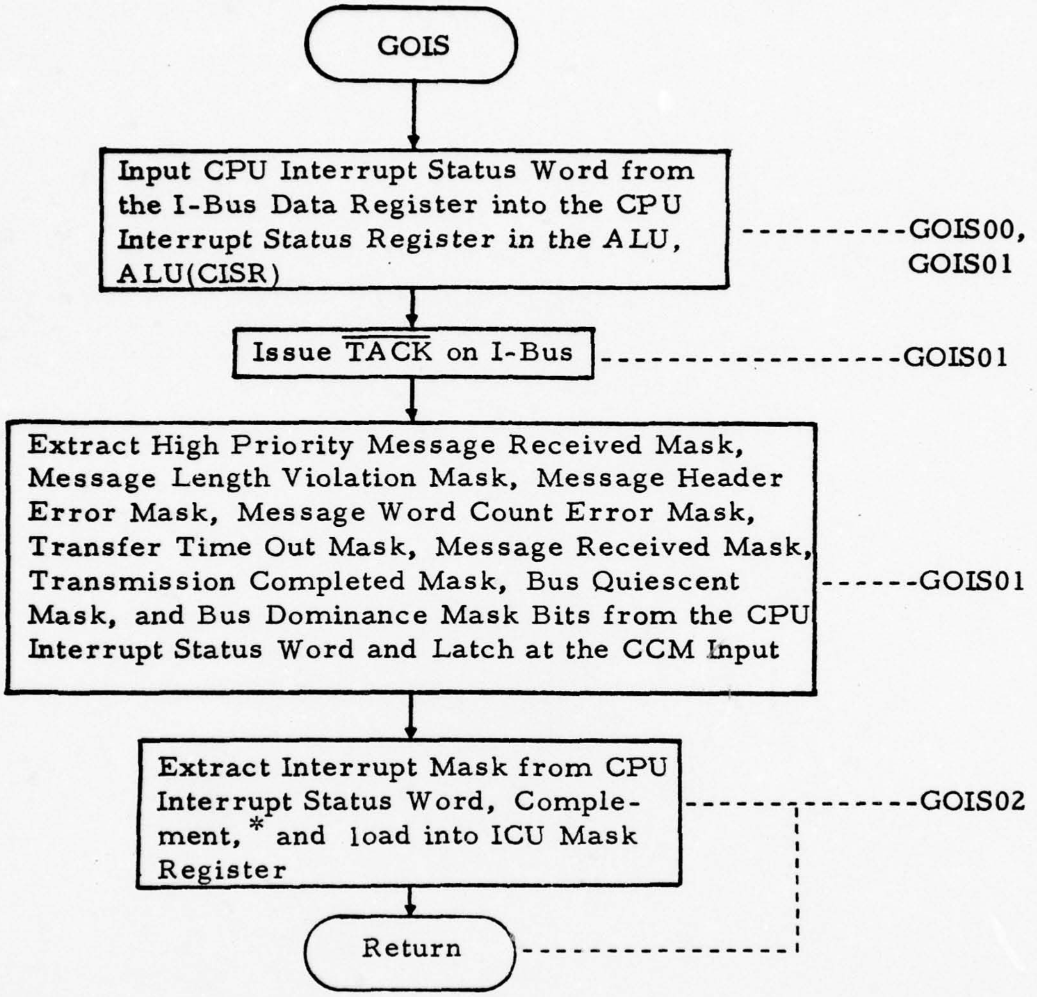
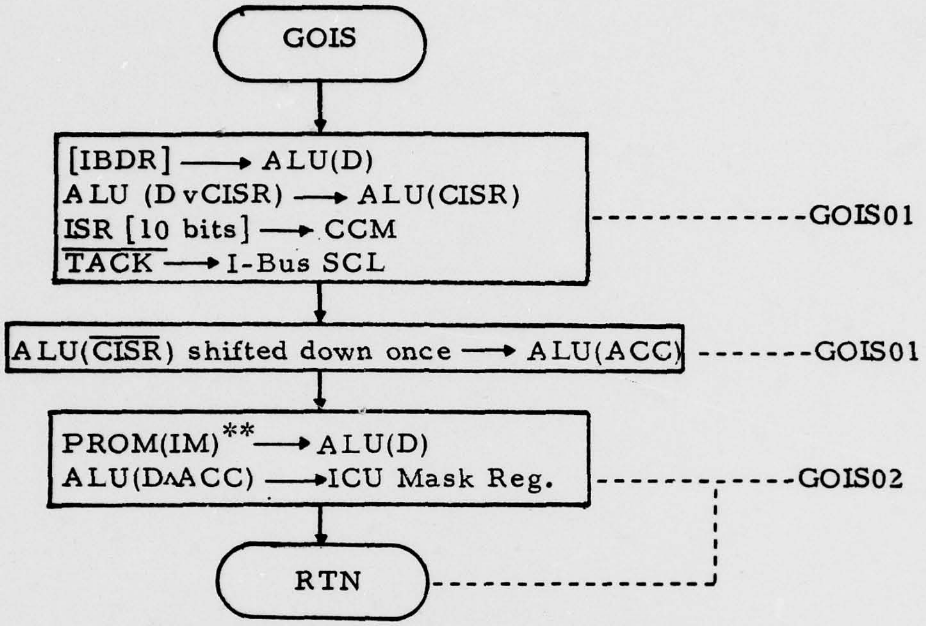

GIA
Clear Last Vector Read Instr → ICU
[CIVB] → IBDR
IA00
[IBDR] → I-Bus
$\overline{\text{IRQCL}}$ → IRQ Flag (I-Bus)
IA01
$\overline{\text{ITACK}}$ → ITACK Flag (I-Bus)
IA02
No
IAK
Yes
$\overline{\text{IBDRCL}}$ → IBDR
$\overline{\text{ITACKCL}}$ → ITACK Flag
IA03
No
CIVBOR
Yes
IC
RTN
IA04

Fig. B-22.B. RTL for Global Interrupt Acknowledge Routine


GOIS
Input CPU Interrupt Status Word from the I-Bus Data Register into the CPU Interrupt Status Register in the ALU, ALU(CISR)
GOIS00, GOIS01
Issue $\overline{\text{TACK}}$ on I-Bus
GOIS01
Extract High Priority Message Received Mask, Message Length Violation Mask, Message Header Error Mask, Message Word Count Error Mask, Transfer Time Out Mask, Message Received Mask, Transmission Completed Mask, Bus Quiescent Mask, and Bus Dominance Mask Bits from the CPU Interrupt Status Word and Latch at the CCM Input
GOIS01
Extract Interrupt Mask from CPU Interrupt Status Word, Complement,* and load into ICU Mask Register
GOIS02
Return

*DP/M Mask (not allow) bit is a 0-bit. Am 2914 mask bit is a 1-bit. Therefore, must complement mask received from PE CPU before loading ICU mask register.

Fig. B-23.A. Functional Flowchart for Global Output Interrupt Status CAW Routine (GOIS)



*The ALU(ISR) contains two bits which are set by the BIU (Specification of the BIU). The CPU Interrupt Status Word contains 0's in these two bit positions.

**IM = Interrupt Mask Word, 0111111111110000.

Fig. B-23.B. RTL for Global Output Interrupt Status CAW Routine

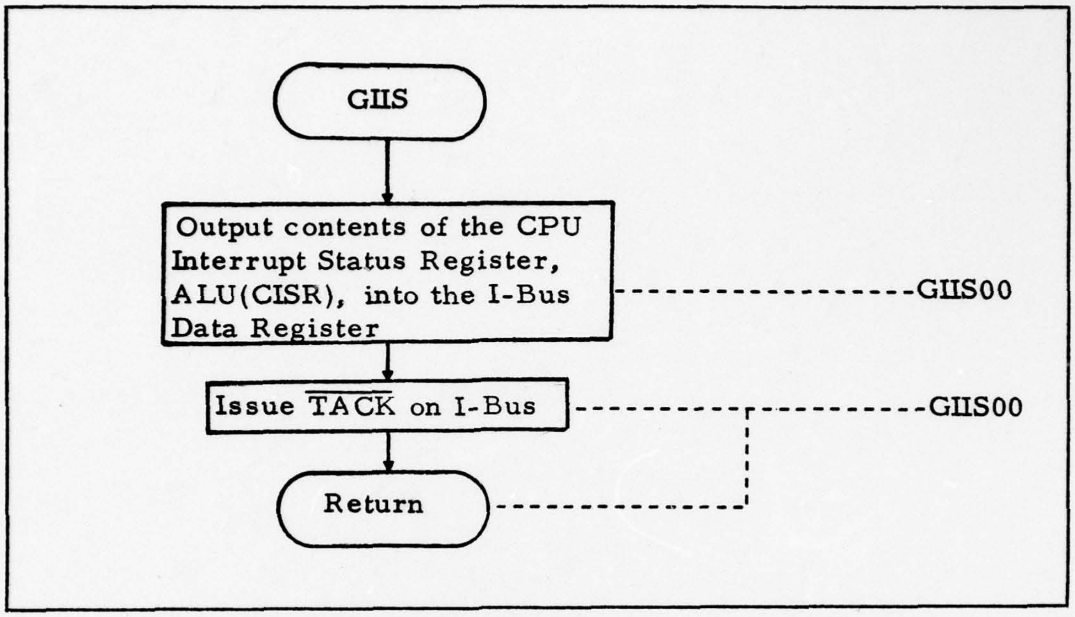

GIIS
Output contents of the CPU Interrupt Status Register, ALU(CISR), into the I-Bus Data Register
GIIS00
Issue $\overline{\text{TACK}}$ on I-Bus
GIIS00
Return

Fig. B-24. A. Functional Flowchart for Global Input Interrupt Status CAW Routine (GIIS)

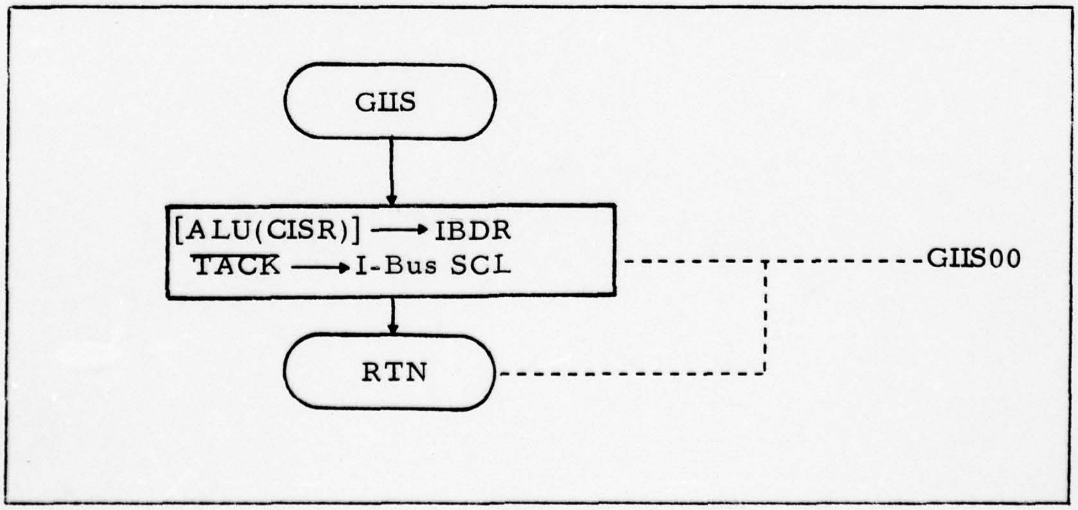


Fig. B-24. B. RTL for Global Input Interrupt Status CAW Routine

AD-A053 269 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 9/2
BUS INTERFACE UNIT DESIGN FOR THE DISTRIBUTED PROCESSOR/MEMORY --ETC(U)
DEC 77 L T KONNO
UNCLASSIFIED AFIT/GE/EE/77-24 NL

3 OF 3
AD
A053269

END
DATE
FILMED
6-78
DDC

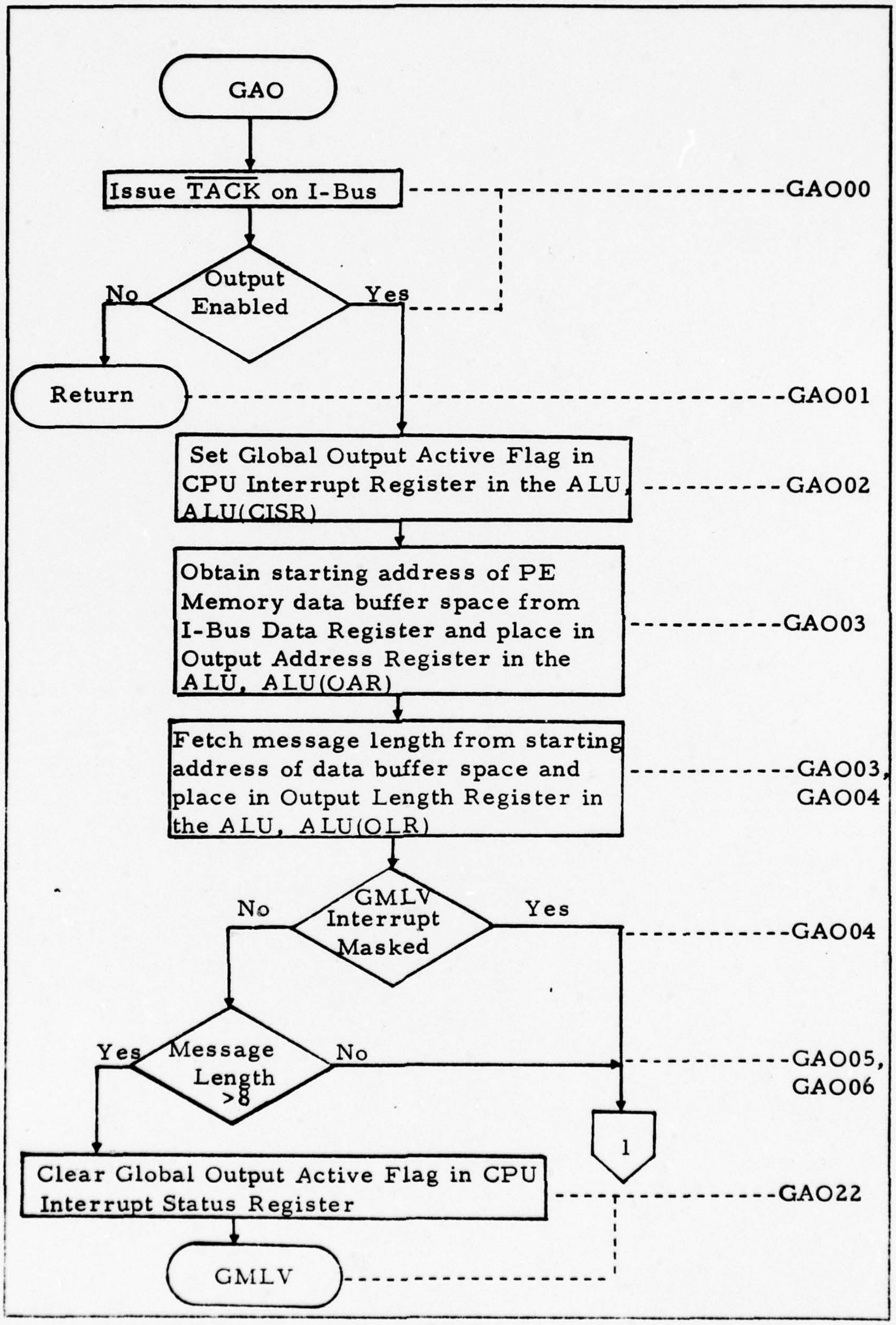
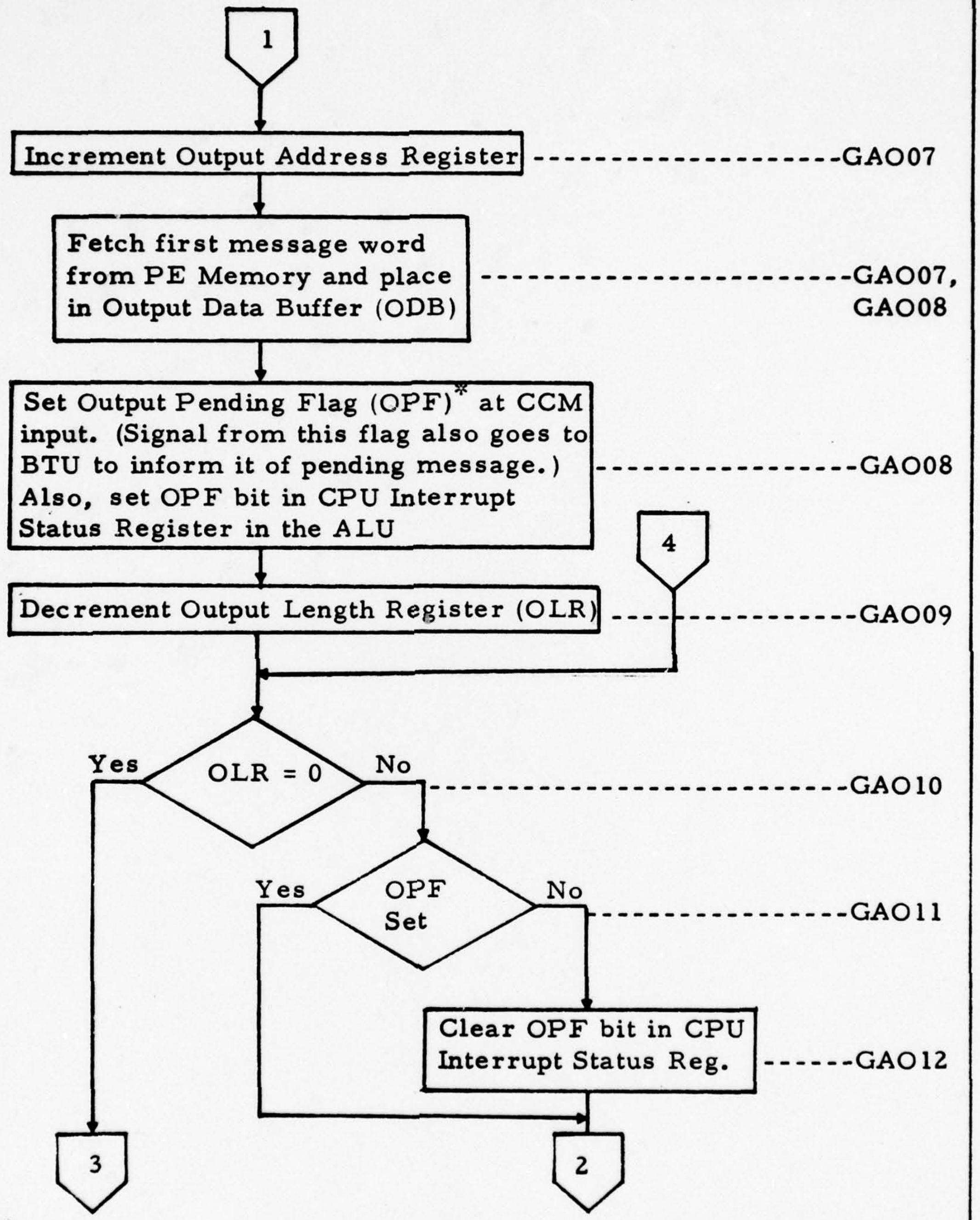

GAO
Issue TACK on I-Bus
GAO00
Output Enabled
No
Yes
Return
GAO01
Set Global Output Active Flag in CPU Interrupt Register in the ALU, ALU(CISR)
GAO02
Obtain starting address of PE Memory data buffer space from I-Bus Data Register and place in Output Address Register in the ALU, ALU(OAR)
GAO03
Fetch message length from starting address of data buffer space and place in Output Length Register in the ALU, ALU(OLR)
GAO03, GAO04
GMLV Interrupt Masked
No
Yes
GAO04
Message Length >8
Yes
No
GAO05, GAO06
1
Clear Global Output Active Flag in CPU Interrupt Status Register
GAO22
GMLV

Fig. B-25.A. Functional Flowchart for Global Activate Output CAW Routine (GAO)


1
Increment Output Address Register
GAO07
Fetch first message word from PE Memory and place in Output Data Buffer (ODB)
GAO07, GAO08
Set Output Pending Flag (OPF)* at CCM input. (Signal from this flag also goes to BTU to inform it of pending message.) Also, set OPF bit in CPU Interrupt Status Register in the ALU
GAO08
4
Decrement Output Length Register (OLR)
GAO09
Yes
OLR = 0
No
GAO10
Yes
OPF Set
No
GAO11
Clear OPF bit in CPU Interrupt Status Reg.
GAO12
3
2

*This flag is normally cleared by a signal from the BTU when it fetches a word from the Output Data Buffer. This routine clears the flag when the output is disabled, and clears the OPF bit in the CPU Interrupt Status Register.

Fig. B-25.A. (continued)

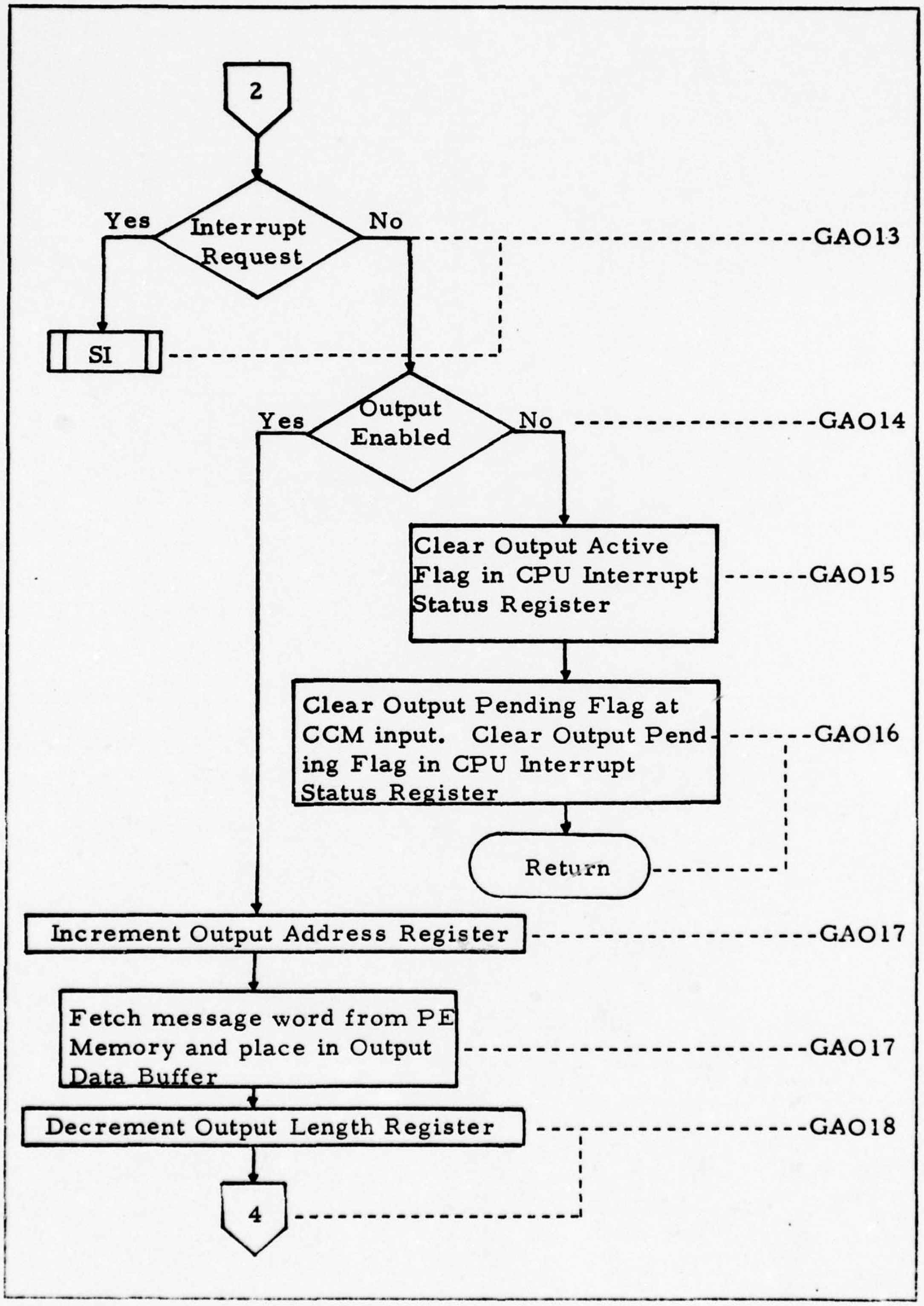

2
Interrupt Request
Yes
No
GAO13
SI
Output Enabled
Yes
No
GAO14
Clear Output Active Flag in CPU Interrupt Status Register
GAO15
Clear Output Pending Flag at CCM input. Clear Output Pending Flag in CPU Interrupt Status Register
GAO16
Return
Increment Output Address Register
GAO17
Fetch message word from PE Memory and place in Output Data Buffer
GAO17
Decrement Output Length Register
GAO18
4

Fig. B-25.A. (continued)

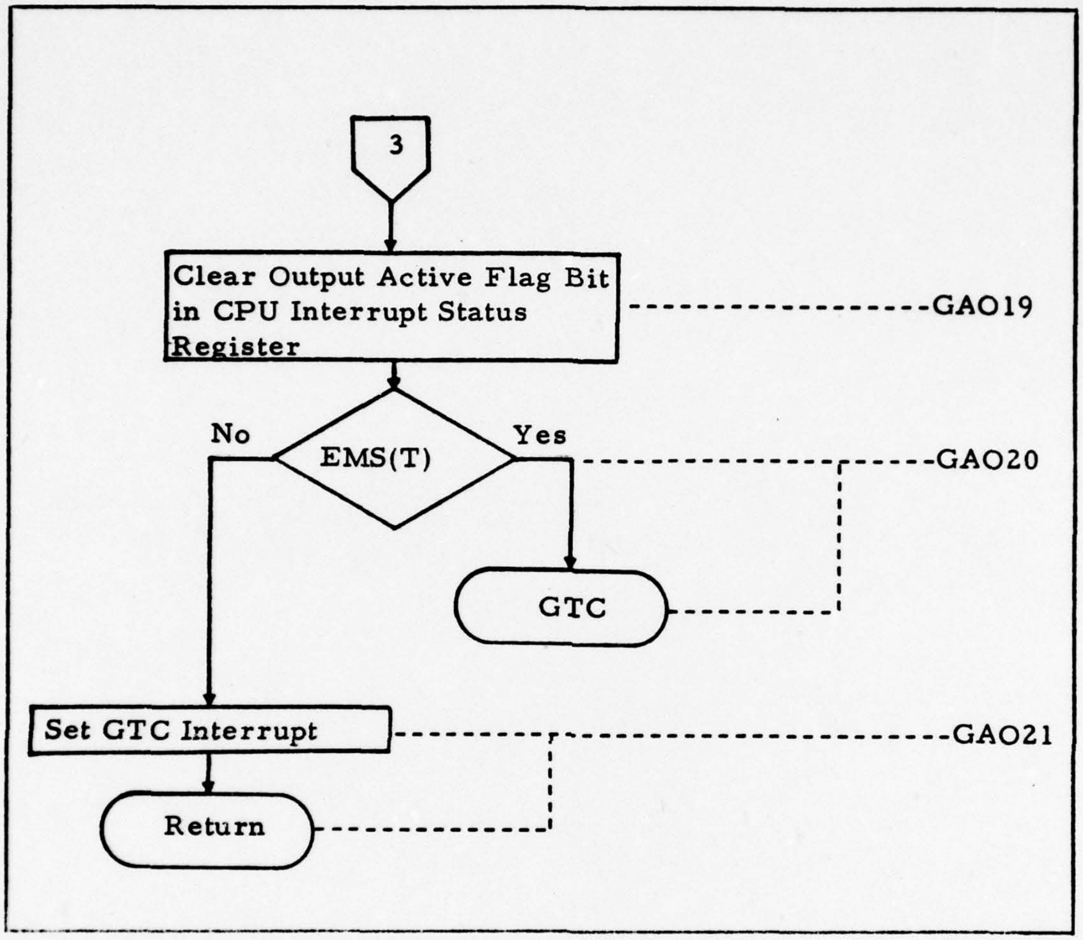
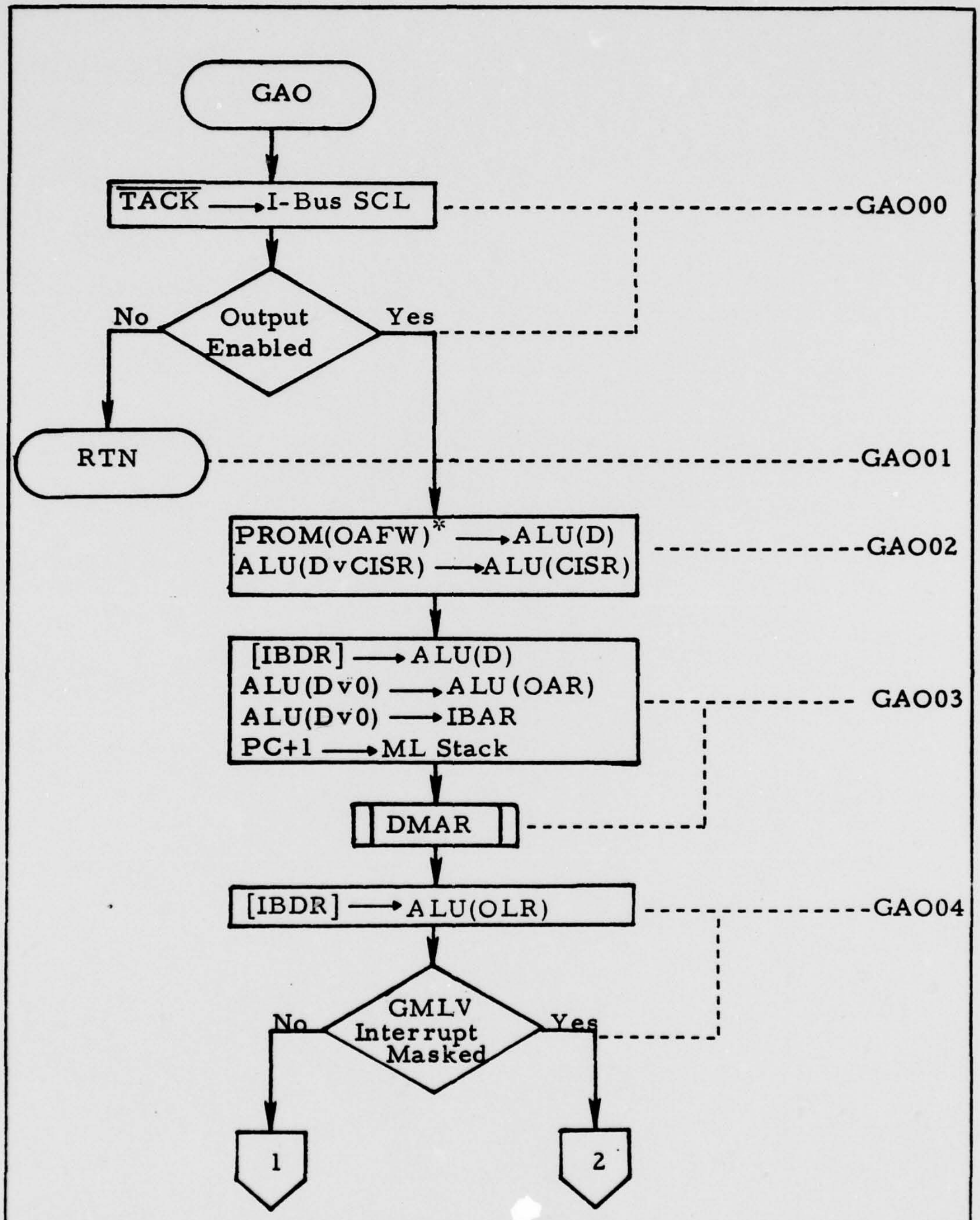
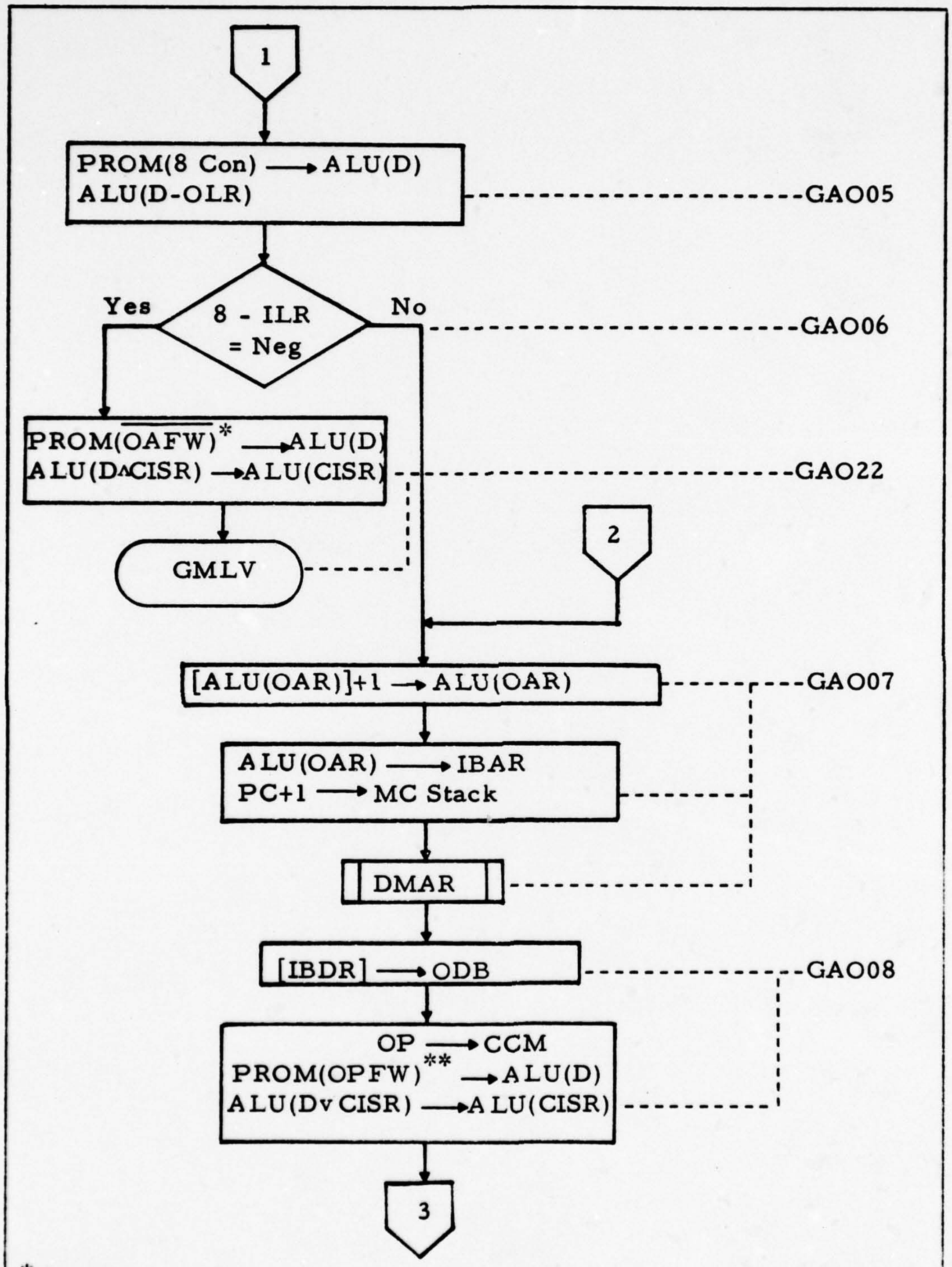

3
Clear Output Active Flag Bit in CPU Interrupt Status Register
GAO19
No
EMS(T)
Yes
GAO20
GTC
Set GTC Interrupt
GAO21
Return

Fig. B-25.A. (continued)



*OAFW = Output Active Flag Word. A 16-bit word with the OAF bit equal to one and the other bits equal to zero.

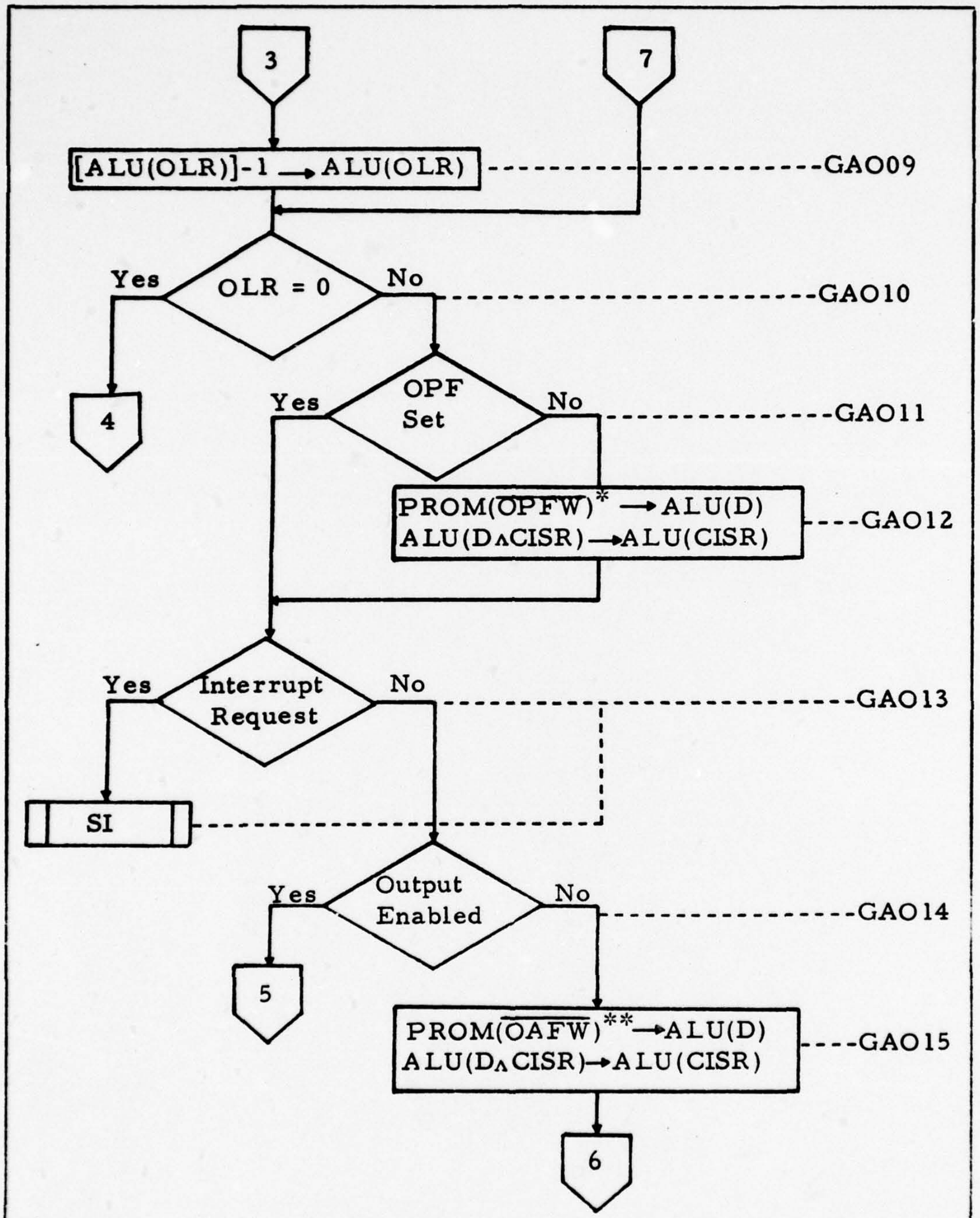
Fig. B-25.B. RTL for Global Activate Output CAW Routine



*$\overline{\text{OAFW}}$ = Clear Output Active Flag Word. A 16-bit word with the OAF bit equal to zero and the other bits equal to one.

**OPFW = Output Pending Flag Word. A 16-bit word with the OPF bit equal to one and the other bits equal to zero.

Fig. B-25.B. (continued)



*$\overline{\text{OPFW}}$ = Clear Output Pending Flag word. A 16-bit word with the OPF bit equal to zero and the other bits equal to one.

**$\overline{\text{OAFW}}$ = Clear Output Active Flag word. A 16-bit word with the OAF bit equal to zero and the other bits equal to one.

Fig. B-25.B. (continued)

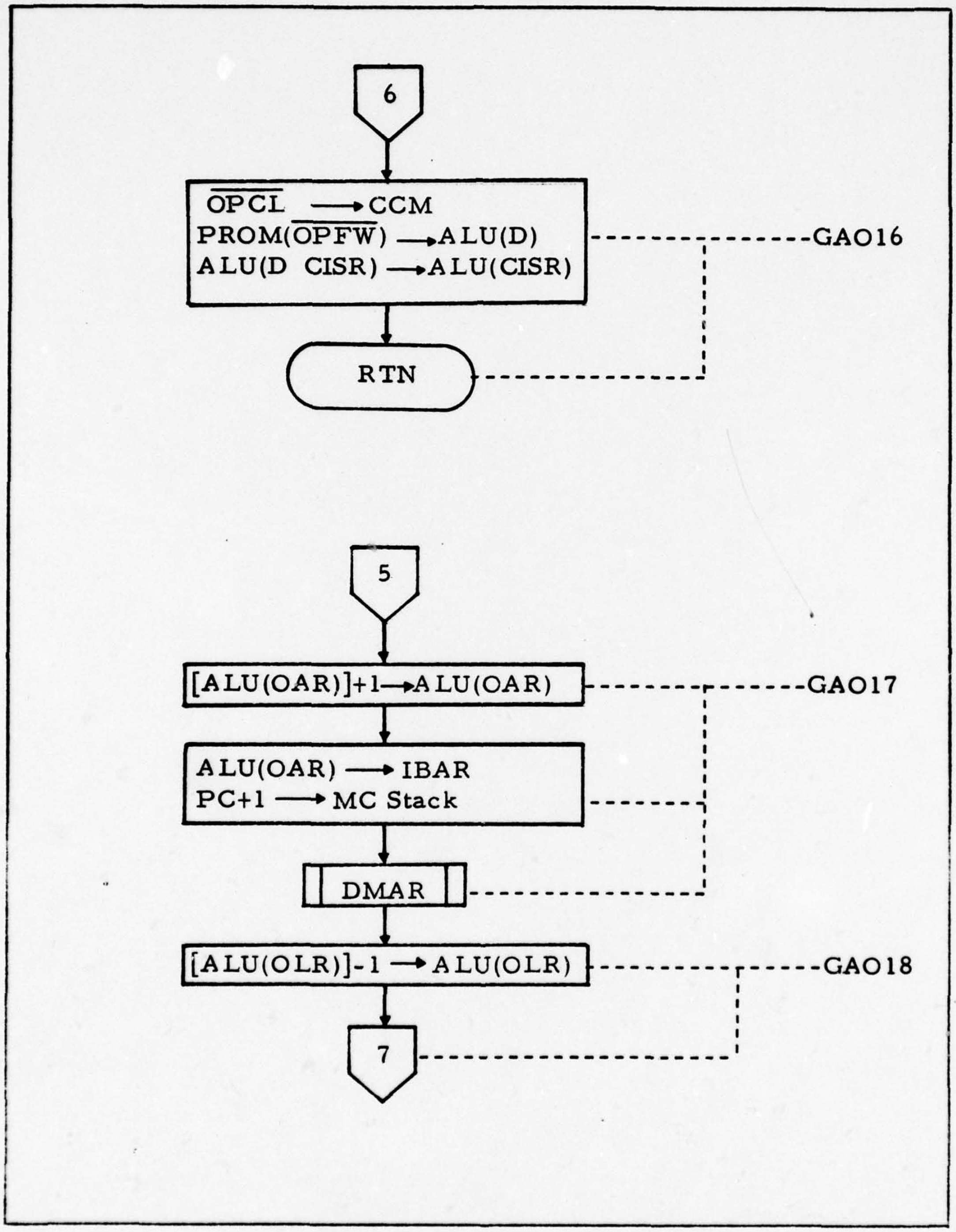


Fig. B-25.B. (continued)

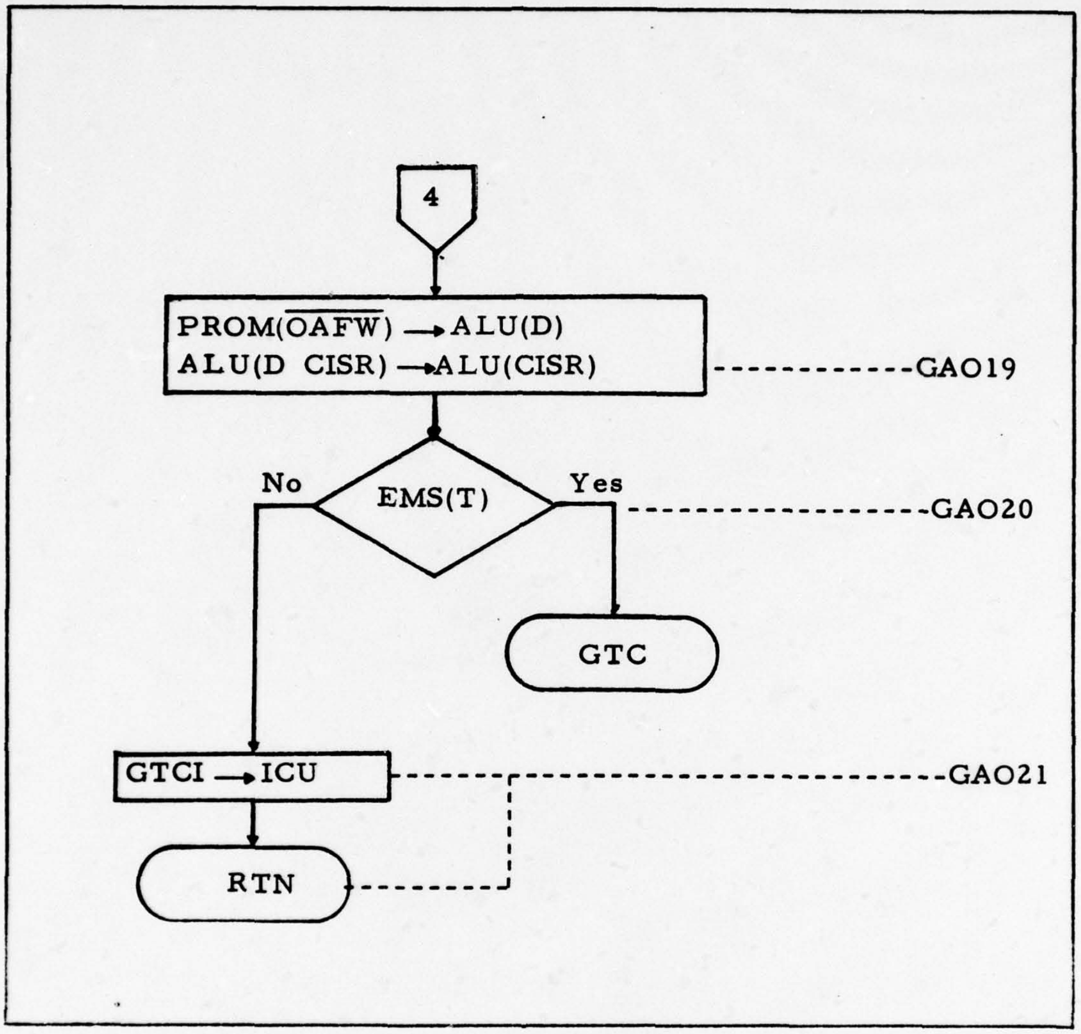


Fig. B-25.B. (continued)

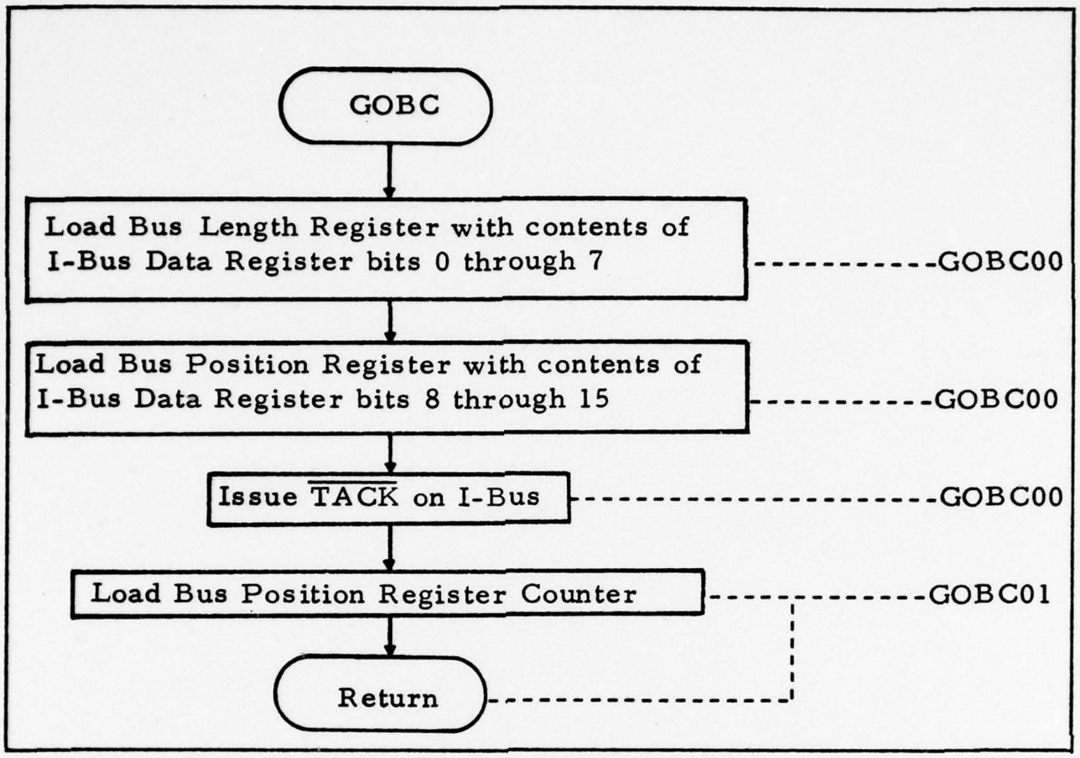

GOBC
Load Bus Length Register with contents of I-Bus Data Register bits 0 through 7
GOBC00
Load Bus Position Register with contents of I-Bus Data Register bits 8 through 15
GOBC00
Issue $\overline{\text{TACK}}$ on I-Bus
GOBC00
Load Bus Position Register Counter
GOBC01
Return

Fig. B-26.A. Functional Flowchart for Global Output Bus Control CAW Routine (GOBC)

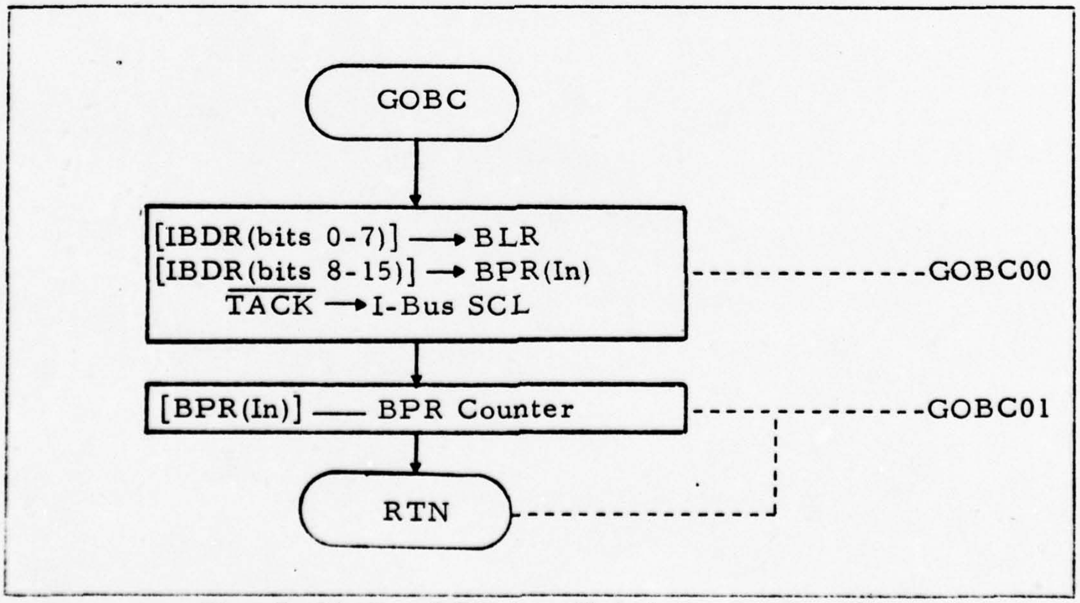


Fig. B-26.B. RTL for Global Output Bus Control CAW Routine

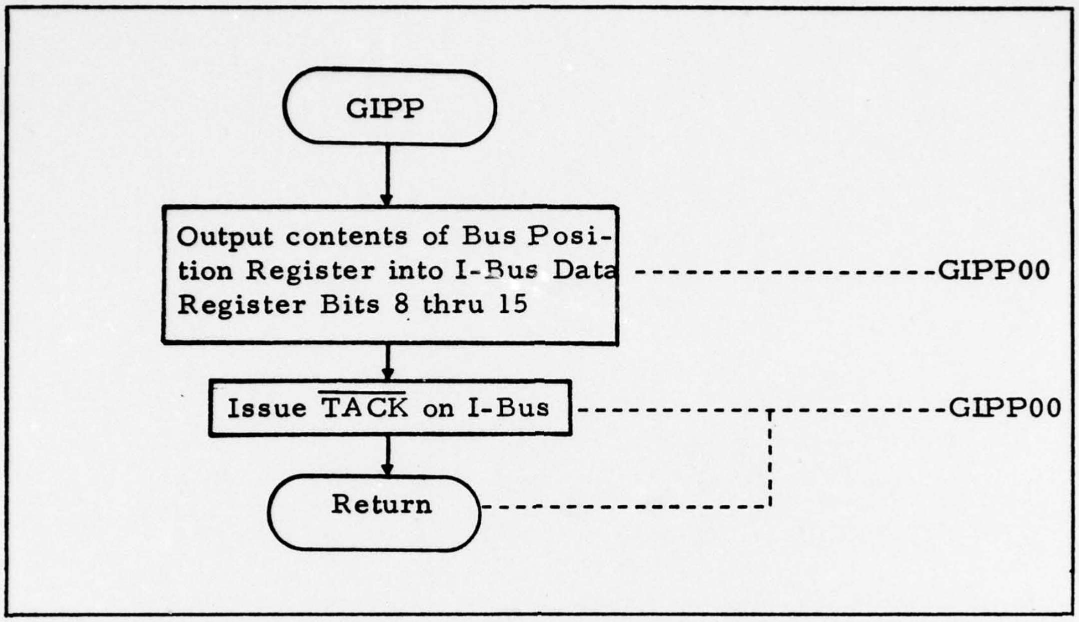

GIPP
Output contents of Bus Position Register into I-Bus Data Register Bits 8 thru 15
GIPP00
Issue $\overline{\text{TACK}}$ on I-Bus
GIPP00
Return

Fig. B-27.A. Functional Flowchart for Global Input Present Position CAW Routine (GIPP)

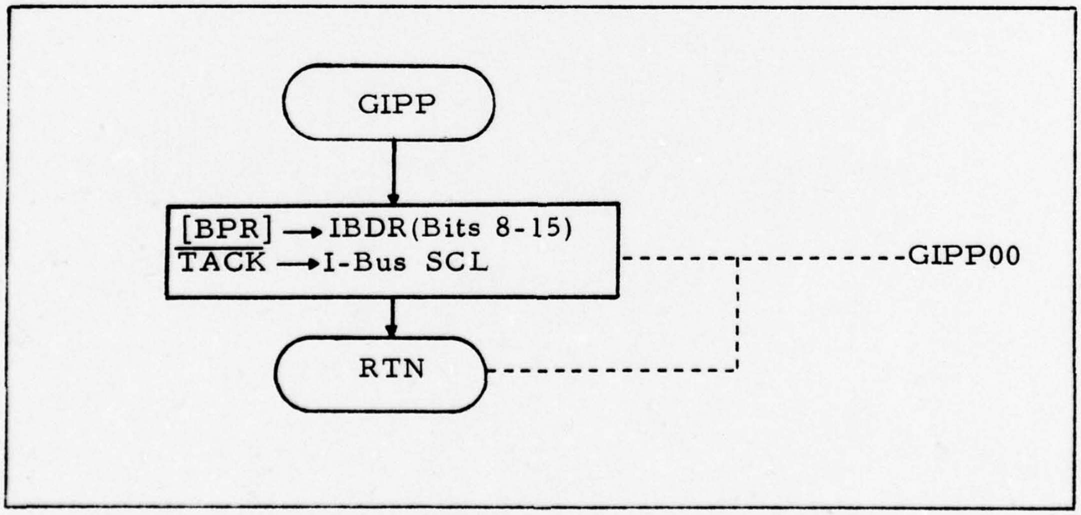


Fig. B-27.B. RTL for Global Input Present Position CAW Routine

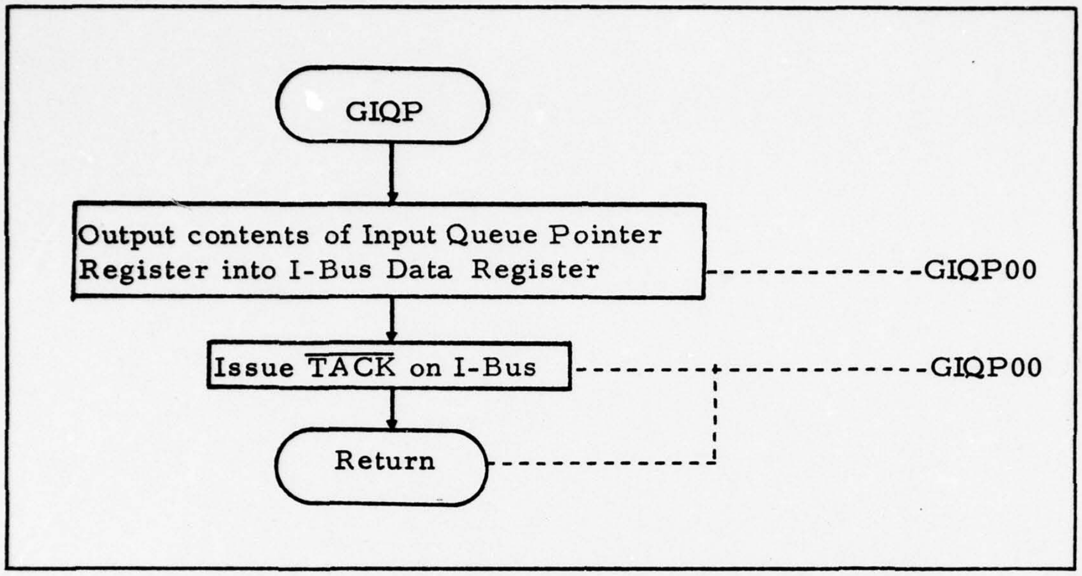

GIQP
Output contents of Input Queue Pointer Register into I-Bus Data Register
GIQP00
Issue $\overline{\text{TACK}}$ on I-Bus
GIQP00
Return

Fig. B-28. A. Functional Flowchart for Global Input Queue Pointer CAW Routine (GIQP)

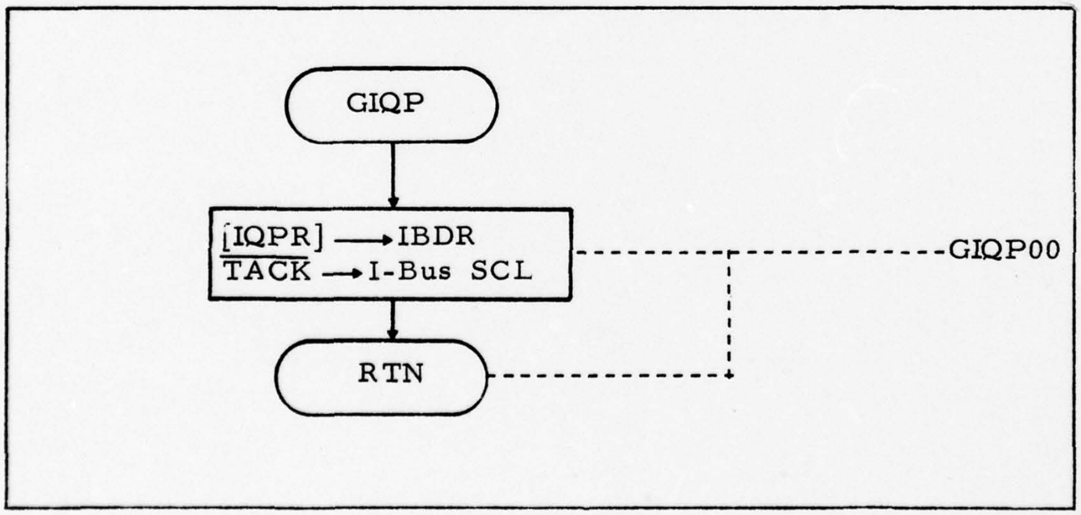


Fig. B-28. B. RTL for Global Input Queue Pointer CAW Routine

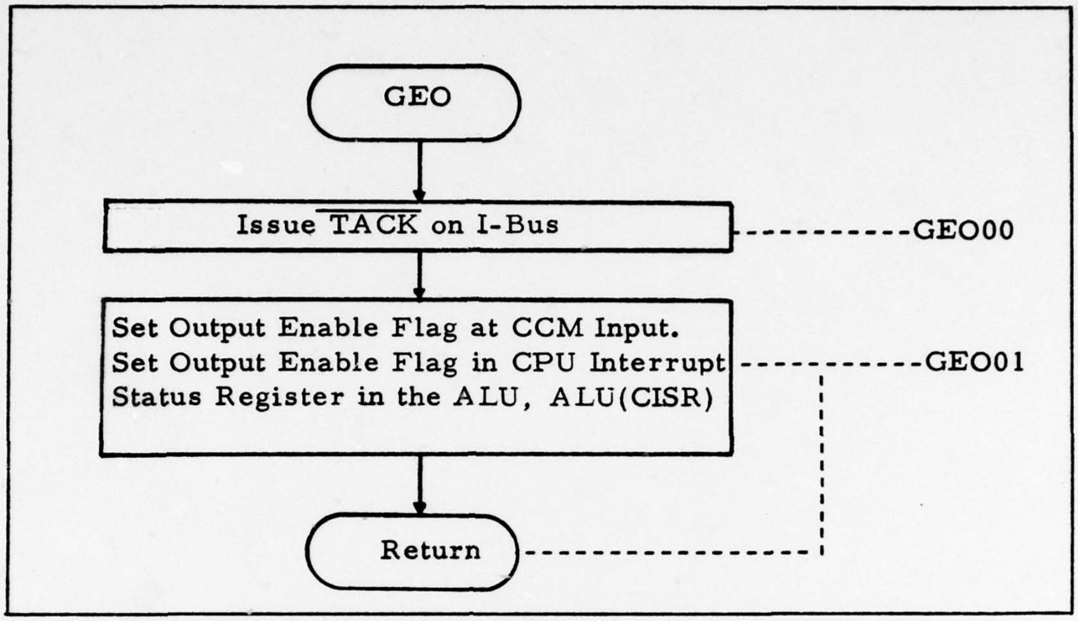
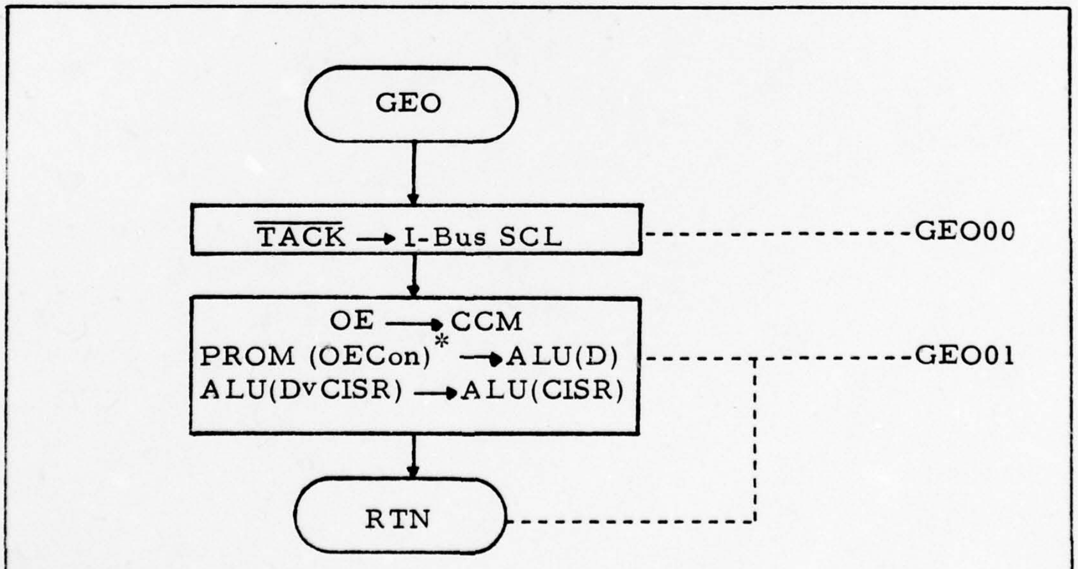

GEO
Issue $\overline{TACK}$ on I-Bus
GEO00
Set Output Enable Flag at CCM Input.
Set Output Enable Flag in CPU Interrupt
Status Register in the ALU, ALU(CISR)
GEO01
Return

Fig. B-29.A. Functional Flowchart for Global Enable Output CAW Routine



*OECon = A 16-bit constant with the Output Enable bit equal to one and the other bits equal to zero.

Fig. B-29.B. RTL for Global Enable Output CAW Routine

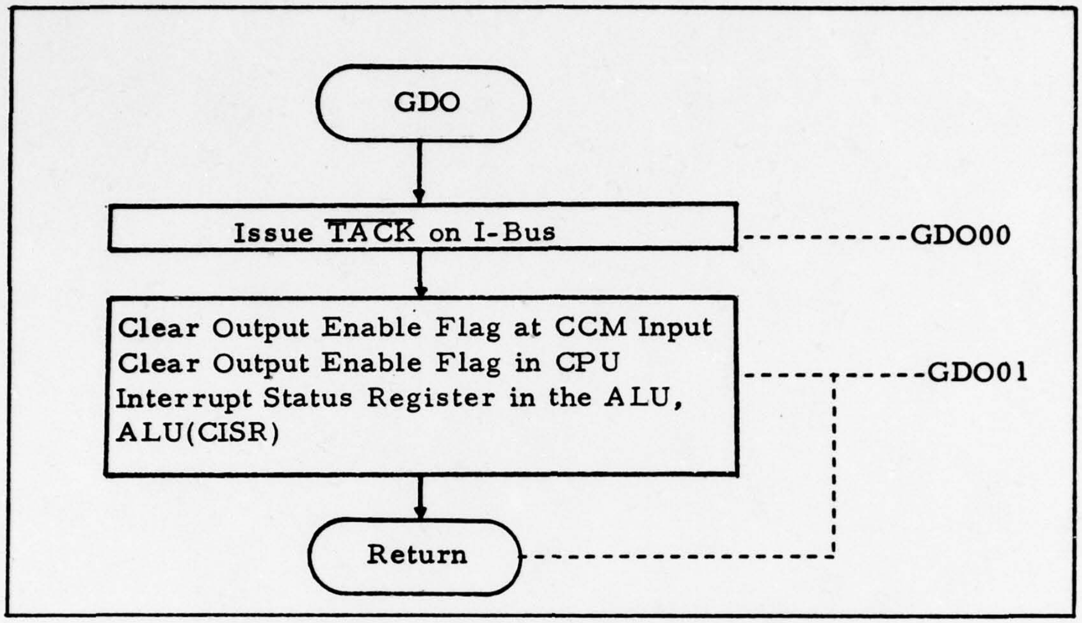
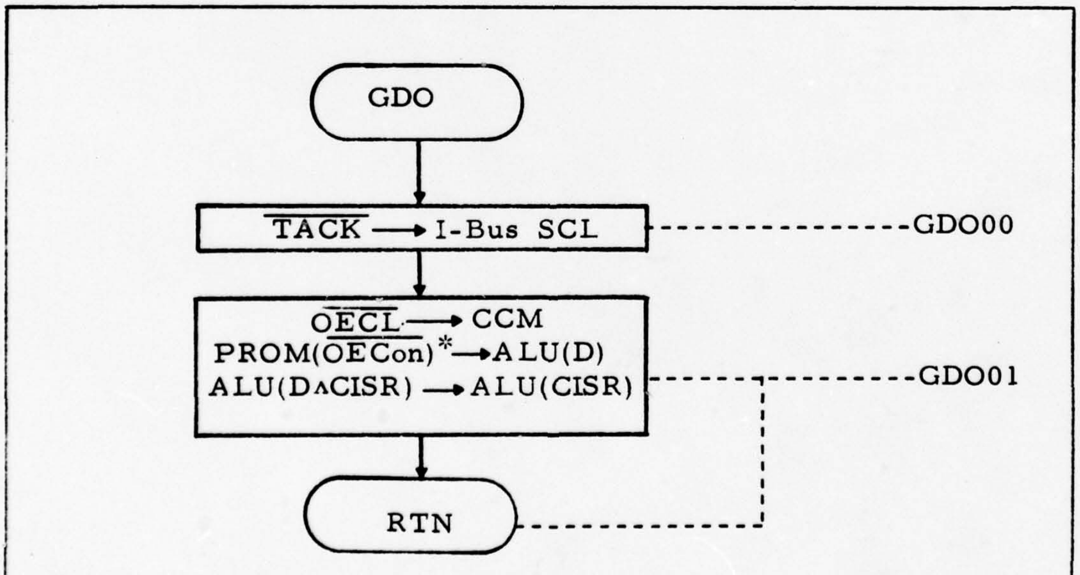

GDO
Issue $\overline{\text{TACK}}$ on I-Bus
GDO00
Clear Output Enable Flag at CCM Input
Clear Output Enable Flag in CPU
Interrupt Status Register in the ALU,
ALU(CISR)
GDO01
Return

Fig. B-30.A. Functional Flowchart for Global Disable Output CAW Routine (GDO)


GDO
$\overline{\text{TACK}} \longrightarrow$ I-Bus SCL
GDO00
$\overline{\text{OECL}} \longrightarrow$ CCM
PROM($\overline{\text{OECon}}$)* $\longrightarrow$ ALU(D)
ALU(D$\wedge$CISR) $\longrightarrow$ ALU(CISR)
GDO01
RTN

*$\overline{\text{OECon}}$ = A 16-bit constant with the Output Enable bit equal to zero and the other bits equal to one.

Fig. B-30.B. RTL for Global Disable Output CAW Routine

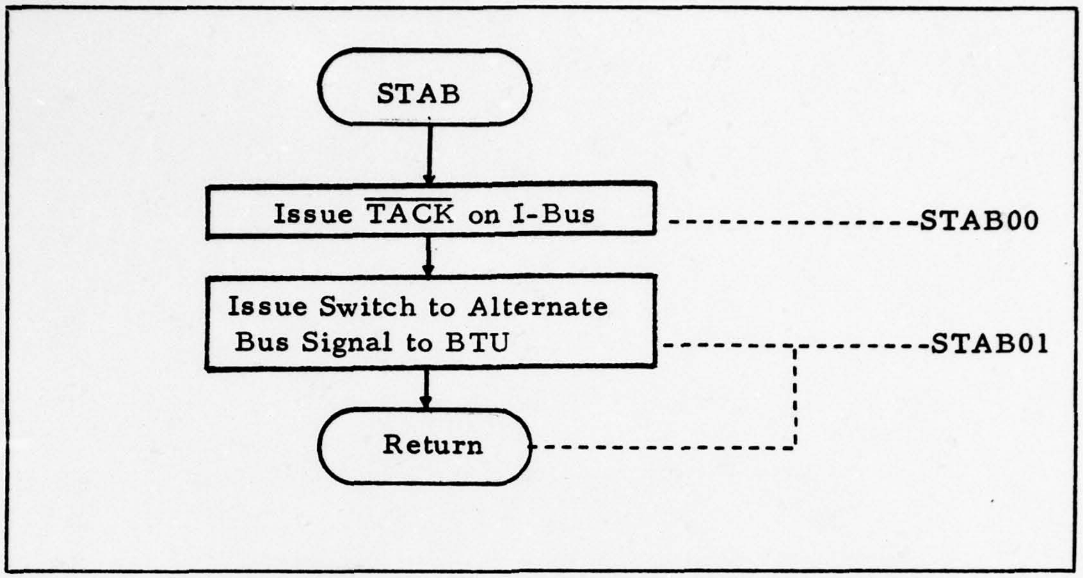

STAB
Issue $\overline{\text{TACK}}$ on I-Bus
STAB00
Issue Switch to Alternate Bus Signal to BTU
STAB01
Return

Fig. B-31.A. Functional Flowchart for Switch to Alternate Bus CAW Routine (STAB)

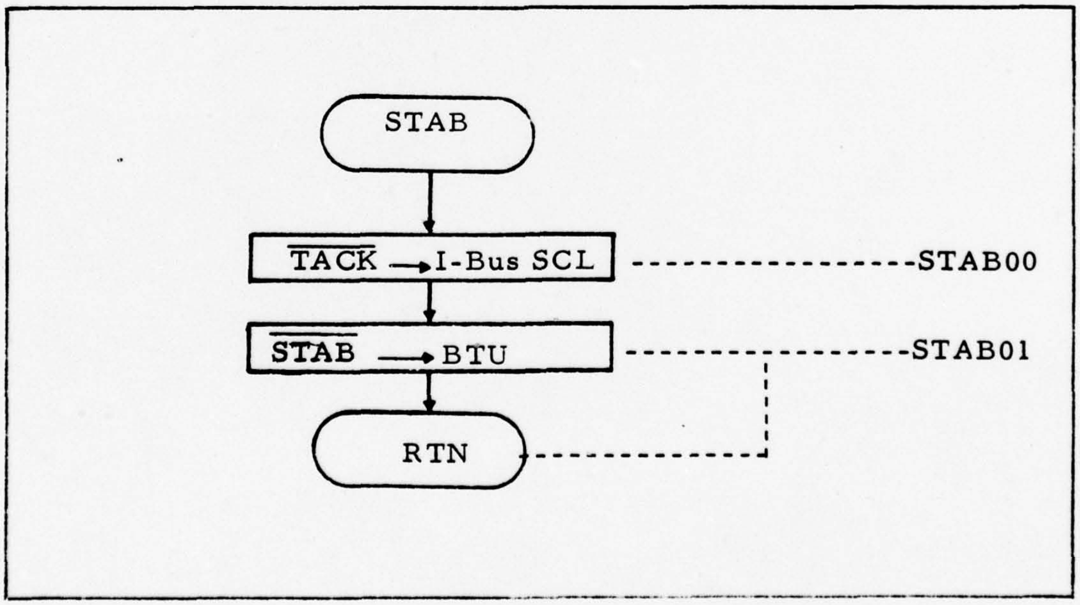

STAB
$\overline{\text{TACK}}$ → I-Bus SCL
STAB00
$\overline{\text{STAB}}$ → BTU
STAB01
RTN

Fig. B-31.B. RTL for Switch to Alternate Bus CAW Routine

Appendix C

BIU Microcodes

This appendix presents the microcodes for the BIU microroutines. The microcodes are organized under the following categories: (1) Microcodes for Service Routines, (2) Microcodes for External Interrupt Routines, (3) Microcodes for Program Generated Interrupt Routines, and (4) Microcodes for CAW Routines.

TABLE C-I

Microcodes for Service Routines

Microroutine	MP Addr	Next Addr Sel Instr	Cond Code Mux Sel	Branch Addr	ALU Source (R/S)	ALU Function	CN	ALU Dest Control	ALU A Addr	ALU B Addr	PROM Addr/ STK Instr	ICU Instr	Set/ Clear	Control
Power Up (PU)	00	CJP	TRUE	WFI00	D/0	R v S	0	Y=F F→B	-	CISR	OCon	MC	-	$\overline{PR9OE}$, $\overline{PROMCS}$
Wait for Interrupt (WFI)	WFI00	CJS	INTR	SI00	-	-	0	-	-	-	-	-	-	$\overline{MCCT}$
	WFI01	CJP	TRUE	WFI00	-	-	0	-	-	-	-	-	-	-
Service Interrupt (SI)	SI00	CONT	-	-	-	-	0	-	-	-	PD	RMR	-	$\overline{PR9OE}$, $\overline{ICUTE}$, STKIEN
	SI01	CONT	-	-	-	-	0	-	-	-	PD	RSR	-	$\overline{PR9OE}$, $\overline{ICUTE}$, STKIEN
	SI02	CJV	TRUE	-	0/B	R v S	0	Y=F	-	CIVR	PD	-	-	$\overline{PR9OE}$, $\overline{ALUOE}$, STKIEN
Return (RTN)	RTN00	CJS	INTR	SI00	-	-	0	-	-	-	-	-	-	-
	RTN01	CONT	-	-	D/0	R v S	0	Y=F F→B	-	CIVR	PS	-	-	$\overline{PR9OE}$, STKIEN, $\overline{STKOE}$
	RTN02	CONT	-	-	D/0	R v S	0	Y=F F→B	-	ACC	PS	-	-	$\overline{PR9OE}$, STKIEN, $\overline{STKOE}$
	RTN03	CONT	-	-	0/B	R v S	0	Y=F	-	ACC	-	LSR	-	$\overline{ALUOE}$, $\overline{ICUTE}$
	RTN04	CONT	-	-	D/0	R v S	0	Y=F F→B	-	ACC	PS	-	-	$\overline{PR9OE}$, STKIEN, $\overline{STKOE}$
	RNT05	CRTN	TRUE	-	0/B	R v S	0	Y=F	-	ACC	-	LMR	-	$\overline{ALUOE}$, $\overline{ICUTE}$
Interrupt CPU (IC)	IC00	CJP	INHB	IC01	-	-	0	-	-	-	-	-	-	$\overline{MCCT}$
	IC01	CJP	TRUE	RTN00	0/A	R v S	0	Y=F	-	-	-	-	$\overline{IRQ}$	$\overline{ALUOE}$, CIVBPL, $\overline{SCDG}$
Direct Memory Access Write (DMAW)	DMAW00	CJP	$\overline{BGI}$	DMAW01	-	-	0	-	-	-	WIBF	-	$\overline{BRQ}$	$\overline{MCCT}$, $\overline{PR9OE}$, $\overline{PROMCS}$, $\overline{IBFRLE}$, $\overline{SCDG}$
	DMAW01	CJP	TACK	DMAW02	-	-	0	-	-	-	-	-	-	$\overline{MCCT}$
	DMAW02	CJP	$\overline{D\ TACK}$	DMAW03	-	-	0	-	-	-	-	-	-	$\overline{MCCT}$, $\overline{IBFRBE}$ $\overline{IBARBE}$, $\overline{IBDRBE}$
	DMAW03	CRTN	TRUE	-	-	-	0	-	-	-	-	-	-	$\overline{IBFRCL}$, $\overline{IBARCL}$, $\overline{IBDRCL}$

TABLE C-I

(continued).

Microroutine	MP Addr	Next Addr Sel Instr	Cond Code Mux Sel	Branch Addr	ALU Source (R/S)	ALU Function	CN	ALU Dest Control	ALU A Addr	ALU B Addr	PROM Addr/ STK Instr	ICU Instr	Set/ Clear	Control
Direct Memory Access Read (DMAR)	DMAR00	CJP	$\overline{\text{BGT}}$	DMAR01	-	-	0	-	-	-	RIBF	-	$\overline{\text{BRQ}}$	$\overline{\text{MCCT}}$, $\overline{\text{PR9OE}}$, $\overline{\text{PROMCS}}$, $\overline{\text{IBFRLE}}$, $\overline{\text{SCDG}}$
	DMAR01	CJP	TACK	DMAR02	-	-	0	-	-	-	-	-	-	$\overline{\text{MCCT}}$
	DMAR02	CJP	$\overline{\text{D TACK}}$	DMAR03	-	-	0	-	-	-	-	-	-	$\overline{\text{MCCT}}$, $\overline{\text{IBFRBE}}$, $\overline{\text{IBARBE}}$
	DMAR03	CRTN	TRUE	-	-	-	0	-	-	-	-	-	-	$\overline{\text{IBDRRLE}}$, $\overline{\text{IBFRCL}}$, $\overline{\text{IBARCL}}$

TABLE C-II

Microcodes for External Interrupt Routines

Microroutine	MP Addr	Next Addr Sel Instr	Cond Code Mux Sel	Branch Addr	ALU Source (R/S)	ALU Function	CN	ALU Dest Control	ALU A Addr	ALU B Addr	PROM Addr/ STK Instr	ICU Instr	Set/ Clear	Control
Message Reception (MR)	MR00	CJP	MH	MR27	D/0	R v S	0	Y=F F→B	-	IDR	-	CILVR	$\overline{WRCL}$	$\overline{ICUTE}$, $\overline{BTUOE}$, $\overline{SCDG}$
	MR01	CJP	GMHE	RTN00	-	-	0	-	-	-	-	-	-	-
	MR02	CJP	RB	MR04	-	-	0	-	-	-	-	-	-	-
	MR03	CJP	TRUE	RTN00	-	-	0	-	-	-	-	-	-	-
	MR04	CJS	INTR	SI00	-	-	0	-	-	-	-	-	-	-
	MR05	CONT	-	-	0/B	R v S	0	Y=F	-	IAR	-	-	-	$\overline{ALUOE}$, $\overline{IBARLE}$
	MR06	CJS	TRUE	DMAW00	0/B	R v S	0	Y=F	-	IDR	-	-	-	$\overline{ALUOE}$, $\overline{IBDRLE}$
	MR07	CONT	-	-	0/B	R + S	1	Y=F F→B	-	IAR	-	-	-	-
	MR08	CONT	-	-	0/B	S - R	0	Y=F F→B	-	ILR	-	-	-	-
	MR09	CJP	ZERO	MR18	-	-	0	-	-	-	-	-	-	-
	MR10	CJP	EMS	MR12	-	-	0	-	-	-	-	-	-	-
	MR11	CJP	TRUE	RTN00	-	-	0	-	-	-	-	-	-	-
	MR12	CJP	BDD	MR15	-	-	0	-	-	-	-	-	-	-
	MR13	CONT	-	-	-	-	0	-	-	-	-	-	$\overline{RBCL}$	$\overline{IQPCP}$, $\overline{SCDG}$
	MR14	CJP	TRUE	GMWCE00	-	-	0	-	-	-	-	-	-	$\overline{IQPCP}$
	MR15	CONT	-	-	D/0	R v S	0	Y=F F→B	-	ACC	BDM	-	$\overline{BDDCL}$	$\overline{PR90E}$, $\overline{PROMCS}$, $\overline{SCDG}$
	MR16	CONT	-	-	D/A	R v S	0	Y=F	ACC	-	PD	RMR	-	$\overline{PR90E}$, $\overline{ICUTE}$, $\overline{ALUOE}$ $\overline{STKIEN}$
	MR17	CJP	TRUE	MR13	-	-	0	-	-	-	PS	LMR	-	$\overline{PR90E}$, $\overline{STKIEN}$, $\overline{STKOE}$ $\overline{ICUTE}$
	MR18	CJP	EMS	MR20	-	-	0	-	-	-	-	-	-	-

TABLE C-II

(continued)

Microroutine	MP Addr	Next Addr Sel Instr	Cond Code Mux Sel	Branch Addr	ALU Source (R/S)	ALU Function	CN	ALU Dest Control	ALU A Addr	ALU B Addr	PROM Addr/ STK Instr	ICU Instr	Set/ Clear	Control
Message Reception (continued)	MR19	CJP	TRUE	MR12	-	-	0	-	-	-	-	-	-	-
	MR20	CJP	BDD	MR24	-	-	0	-	-	-	-	-	$\overline{\text{EMSCL}}$	$\overline{\text{SCDG}}$
	MR21	CONT	-	-	-	-	0	-	-	-	-	-	$\overline{\text{RBCL}}$	$\overline{\text{SCDG}}$, $\overline{\text{IQPCP}}$
	MR22	CJP	HP	GHPMR00	-	-	0	-	-	-	-	-	-	-
	MR23	CJP	TRUE	GMHR00	-	-	0	-	-	-	-	-	-	-
	MR24	CONT	-	-	D/0	R v S	0	Y=F F→B	-	ACC	$\overline{\text{BDM}}$	-	$\overline{\text{BDDCL}}$	$\overline{\text{PR90E}}$, $\overline{\text{PROMCS}}$, $\overline{\text{SCDG}}$
	MR25	CONT	-	-	D/A	R ∧ S	0	Y=F	ACC	-	PD	RMR	-	$\overline{\text{PR90E}}$, $\overline{\text{ICUIE}}$, $\overline{\text{ALUOE}}$, $\overline{\text{STKIEN}}$
	MR26	CJP	TRUE	MR21	-	-	0	-	-	-	PS	LMR	-	$\overline{\text{PR90E}}$, $\overline{\text{STKIEN}}$, $\overline{\text{STKOE}}$, $\overline{\text{ICUIE}}$
	MR27	CJS	INTR	S100	-	-	0	-	-	-	-	-	$\overline{\text{MHCL}}$	$\overline{\text{SCDG}}$
	MR28	CJP	GMHE	RTN00	-	-	0	-	-	-	-	-	-	-
	MR29	LDCT	-	03_{16}	D/A	R v S	0	Y=F F→B	IDR	ACC	-	-	-	$\overline{\text{ALUOE}}$, $\overline{\text{RBRG}}$
	MR30	RPCT	-	MR30	D/B	R v S	0	F/2→B	ACC	ACC	-	-	-	-
	MR31	CONT	-	-	D/A	R ∧ S	0	Y=F F→B	ACC	ACC	MIDZC	-	-	$\overline{\text{PR90E}}$, $\overline{\text{PROMCS}}$
	MR32	CONT	-	-	D/A	R v S	0	Y=F	ACC	-	MIDC	-	-	$\overline{\text{PR90E}}$, $\overline{\text{PROMCS}}$, $\overline{\text{ALUOE}}$, $\overline{\text{PARLE}}$
	MR33	CONT	-	-	D/0	R v S	0	Y=F F→B	-	ACC	-	-	-	$\overline{\text{PAROE}}$, $\overline{\text{PROMCS}}$
	MR34	CONT	-	-	D/A	R ∧ S	0	Y=F	ACC	-	-	-	-	$\overline{\text{RBRDOE}}$
	MR35	CJP	$\overline{\text{ZERO}}$	MR37	-	-	0	-	-	-	-	-	-	-

TABLE C-II

(continued)

Microroutine	MP Addr	Next Addr Sel Instr	Cond Code Mux Sel	Branch Addr	ALU Source (R/S)	ALU Function	CN	ALU Dest Control	ALU A Addr	ALU B Addr	PROM Addr/ STK Instr	ICU Instr	Set/ Clear	Control
Message Reception (continued)	MR36	CJP	TRUE	RTN00	-	-	0	-	-	-	-	-	-	-
	MR37	CONT	-	-	0/B	R v S	0	Y=F	-	IDR	-	-	RB	$\overline{\text{IBDRLE}}$, $\overline{\text{SCDG}}$
	MR38	CJS	TRUE	DMAW00	-	-	0	-	-	-	-	-	-	$\overline{\text{IQPROE}}$, $\overline{\text{IBARLE}}$
	MR39	CONT	-	-	D/A	R ∧ S	0	Y=F F→B	IDR	IAR	MBZC	-	-	$\overline{\text{PR9OE}}$, $\overline{\text{PROMCS}}$
	MR40	CJS	TRUE	DMAR00	D/A	R v S	0	Y=F	IAR	-	MBC	-	-	$\overline{\text{PR9OE}}$, $\overline{\text{PROMCS}}$, $\overline{\text{ALUOE}}$ $\overline{\text{IBARLE}}$
	MR41	CONT	-	-	D/0	R v S	0	Y=F F→B	-	IAR	-	-	-	$\overline{\text{IBDROE}}$
	MR42	CJS	TRUE	DMAR00	0/B	R v S	0	Y=F	-	IAR	-	-	-	$\overline{\text{ALUOE}}$, $\overline{\text{IBARLE}}$
	MR43	CONT	-	-	D/0	R v S	0	Y=F F→B	-	ILR	-	-	-	$\overline{\text{IBDROE}}$
	MR44	CONT	-	-	0/B	R + S	1	Y=F F→B	-	IAR	-	-	-	-
	MR45	CONT	-	-	D/A	R - S	0	Y=F	ILR	-	8Con	-	-	$\overline{\text{PR9OE}}$, $\overline{\text{PROMCS}}$
	MR46	CP	NEG	MR51	-	-	0	-	-	-	-	-	-	-
	MR47	CONT	-	-	0/B	R + S	0	Y=F	-	IDR	-	-	-	-
	MR48	CJP	NEG	MR50	-	-	0	-	-	-	-	-	-	-
	MR49	CJP	TRUE	RTN00	-	-	0	-	-	-	-	-	-	-
	MR50	CJP	TRUE	RTN00	-	-	0	-	-	-	-	-	HP	$\overline{\text{SCDG}}$
	MR51	CONT	-	-	D/0	R v S	0	Y=F F→B	-	ACC	BDM	-	-	$\overline{\text{PR9OE}}$, $\overline{\text{PROMCS}}$
	MR52	CONT	-	-	D/A	R v S	0	Y=F	ACC	-	PD	RMR	BDD	$\overline{\text{PR9OE}}$, $\overline{\text{ICUIE}}$, $\overline{\text{ALUOE}}$, $\overline{\text{STKIEN}}$, $\overline{\text{SCDG}}$
	MR53	CJP	TRUE	MR47	-	-	0	-	-	-	PS	LMR	-	$\overline{\text{PR9OE}}$, $\overline{\text{STKIEN}}$, $\overline{\text{STKOE}}$, $\overline{\text{ICUIE}}$

TABLE C-II

(continued)

Microroutine	MP Addr	Next Addr Sel Instr	Cond Code Mux Sel	Branch Addr	ALU Source (R/S)	ALU Function	CN	ALU Dest Control	ALU A Addr	ALU B Addr	PROM Addr/ STK Instr	ICU Instr	Set/ Clear	Control
Global Bus Quiescent (GBQ)	GBQ00	CJP	TRUE	IC00	D/0	R v S	0	Y=F F→B	-	CIVR	GBQIV	CILVR	-	$\overline{ICUTE}$, $\overline{PR9OE}$, $\overline{PROMCS}$
Global Bus Dominance (GBD)	GBD00	CJP	TRUE	IC00	D/0	R v S	0	Y=F F→B	-	CIVR	GBDIV	CILVR	-	$\overline{ICUTE}$, $\overline{PR9OE}$, $\overline{PROMCS}$
Global Message Word Length Error (GMWLE)	GMWLE00	CJP	MH	GMIE00	-	-	0	-	-	-	-	CILVR	$\overline{CMWLECL}$	SCDG, $\overline{ICUTE}$
	GMWLE01	CJP	TRUE	IC00	D/0	R v S	0	Y=F F→B	-	CIVR	GMWLEIV	-	-	$\overline{PR9OE}$, $\overline{PROMCS}$
Global Message Parity Error (GMPE)	GMPE00	CJP	MI	GMIE00	-	-	0	-	-	-	-	CILVR	$\overline{GMPECL}$	SCDG, $\overline{ICUTE}$
	GMPE01	CJP	TRUE	IC00	D/0	R v S	0	Y=F F→B	-	CIVR	GMPEIV	-	-	$\overline{PR9OE}$, $\overline{PROMCS}$
Global Message Encoding Error (GMEE)	GMEE00	CJP	MH	GMIE00	-	-	0	-	-	-	-	CILVR	$\overline{GMEECL}$	SCDG, $\overline{ICUTE}$
	GMEE01	CJP	TRUE	IC00	D/0	R v S	0	Y=F F→B	-	CIVR	GMEEIV	-	-	$\overline{PR9OE}$, $\overline{PROMCS}$
Global Transfer Time Out (GTTO & GTTOI)	GTT000	CJP	GTTOM	GTT002	-	-	0	-	-	-	-	-	-	-
	GTT001	CJP	TRUE	IC00	D/0	R v S	0	Y=F F→B	-	CIVR	GTTOIV	-	-	$\overline{PR9OE}$, $\overline{PROMCS}$
	GTT002	CJP	TRUE	RTN00	-	-	0	-	-	-	-	-	$\overline{GTTOI}$	$\overline{SCDG}$
	GTTOI00	CJP	TRUE	GTT001	-	-	0	-	-	-	-	CILVR	-	$\overline{ICUTE}$
Global CAM Interrupt (GCI)	GCI00	JMAP	-	-	-	-	0	-	-	-	-	CILVR	-	$\overline{ICUTE}$

TABLE C-III

Microcodes for Program Generated Interrupt Routines

Microroutine	MP Addr	Next Addr Sel Instr	Cond Code Mux Sel	Branch Addr	ALU Source (R/S)	ALU Function	CN	ALU Dest Control	ALU A Addr	ALU B Addr	PROM Addr/ STK Instr	ICU Instr	Set/ Clear	Control
Global High Priority Message Received (GHPMR & GHPMRI)	GHPMR00	CJP	GHPMRM	GHPMR02	-	-	0	-	-			-	$\overline{HPCL}$	$\overline{SCDG}$
	GHPMR01	CJP	TRUE	IC00	D/0	R v S	0	Y=F F→B	-	CIVR	GHPMRIV	-	-	$\overline{PR9OE}$, $\overline{PROMCS}$
	GHPMR02	CJP	TRUE	RTN00	-	-	0	-	-	-	-	-	GHPMRT	$\overline{SCDG}$
	GHPMRI00	CJP	TRUE	GHPMR01	-	-	0	-	-	-	-	CILVR	-	$\overline{ICUIE}$
Global Message Length Violation (GMLV & GMLVI)	GMLV00	CJP	GMLVM	GMLV02	-	-	0	-	-	-	-	-	-	-
	GMLV01	CJP	TRUE	IC00	D/0	R v S	0	Y=F F→B	-	CIVR	GMLVIV	-	-	$\overline{PR9OE}$, $\overline{PROMCS}$
	GMLV02	CJP	TRUE	RTN00	-	-	0	-	-	-	-	-	$\overline{GMLVI}$	$\overline{SCDG}$
	GMLVI00	CJP	TRUE	GMLV01	-	-	0	-	-	-	-	CILVR	-	$\overline{ICUIE}$
Global Message Header Error (GMHE & GMHEI)	GMHE00	CJP	GMHEM	GMHE02	-	-	0	-	-	-	-	-	GMHE	$\overline{SCDG}$
	GMHE01	CJP	TRUE	IC00	D/0	R v S	0	Y=F F→B	-	CIVR	GMHEIV	-	-	$\overline{PR9OE}$, $\overline{PROMCS}$
	GMHE02	CJP	TRUE	RTN00	-	-	0	-	-	-	-	-	$\overline{GMHET}$	$\overline{SCDG}$
	GMHEI00	CJP	TRUE	GMHE01	-	-	0	-	-	-	-	CILVR	-	$\overline{ICUIE}$
Global Message Word Count Error (GMWCE & GMWCEI)	GMWCE00	CJP	GMWCEM	GMWCE02	-	-	0	-	-	-	-	-	-	-
	GMWCE01	CJP	TRUE	IC00	D/0	R v S	0	Y=F F→B	-	CIVR	GMWCEIV	-	-	$\overline{PR9OE}$, $\overline{PROMCS}$
	GMWCE02	CJP	TRUE	RTN00	-	-	0	-	-	-	-	-	$\overline{GMWCEI}$	$\overline{SCDG}$
	GMWCEI00	CJP	TRUE	GMWCE01	-	-	0	-	-	-	-	CILVR	-	$\overline{ICUIE}$
Global Message Received (GMR)	GMR00	CJP	GMRM	RTN00	-	-	0	-	-	-	-	-	-	-
	GMR01	CJP	TRUE	IC00	D/0	R v S	0	Y=F F→B	-	CIVR	GMRIV	-	-	$\overline{PR9OE}$, $\overline{PROMCS}$

TABLE C-III

(continued)

Microroutine	MP Addr	Next Addr Sel Instr	Cond Code Mux Sel	Branch Addr	ALU Source (R/S)	ALU Function	CN	ALU Dest Control	ALU A Addr	ALU B Addr	PROM Addr/ STK Instr	ICU Instr	Set/ Clear	Control
Global Transmission Completed (GTC & GTCI)	GTC00	CJP	GTCM	GTC02	-	-	0	-		-	-	-	$\overline{EMS(T)CL}$	$\overline{SCDG}$
	GTC01	CJP	TRUE	IC00		R v S	0	Y=F F B	-	CIVR	GTCIV	-	-	$\overline{PR9OE}$, $\overline{PROMCS}$
	GTC02	CJP	TRUE	RTN00	-	-	0	-	-	-	-	- .	$\overline{GTCI}$	$\overline{SCDG}$
	GTCI00	CJP	TRUE	GTC01	-	-	0	-	-	-	-	CILVR	$\overline{GTCICL}$	$\overline{ICUIE}$, $\overline{SCDG}$
Global Interrupt Acknowledge (GIA)	GIA00	-	-	-	-	-	0	-	-	-	-	CILVR	-	$\overline{ICUIE}$, CIVBPD, $\overline{IBDRLE}$
	GIA01	-	-	-	-	-	0	-	-	-	-	-	$\overline{IRQCL}$	$\overline{IBDRBE}$, $\overline{SCDG}$
	GIA02	CJP	IAK	IA03	-	-	0	-	-	-	-	-	$\overline{ITACK}$	SCDG, $\overline{MCCT}$
	GIA03	CJP	CIVBOR	IC00	-	-	0	-	-	-	-	-	$\overline{ITACKCL}$	SCDG, $\overline{IBDRCL}$
	GIA04	CJP	TRUE	RTH00	-	-	0	-	-	-	-	-	-	-

TABLE C-IV

Microcodes for CAW Routines

Microroutine	MP Addr	Next Addr Sel Instr	Cond Code Mux Sel	Branch Addr	ALU Source (R/S)	ALU Function	CN	ALU Dest Control	ALU A Addr	ALU B Addr	PROM Addr/ STK Instr	ICU Instr	Set/ Clear	Control
Global Output Interrupt Status (GOIS)	GOIS00	CONT	-	-	D/A	$R \vee S$	0	Y=F F→B	CISR	CISR	-	-	TACK	$\overline{\text{IBDROE}}$, $\overline{\text{ALUOE}}$, $\overline{\text{CCLG}}$ $\overline{\text{SCDG}}$
	GOIS01	CONT	-	-	0/A	$\overline{R \vee S}$	0	Y=F F/2→B	CISR	ACC	-	-	-	-
	GOIS02	CJP	TRUE	RTN00	D/A	$R \wedge S$	0	Y=F	ACC	-	IM	LMR	-	$\overline{\text{PR90E}}$, $\overline{\text{PROMCS}}$, $\overline{\text{ALUOE}}$, $\overline{\text{ICUIE}}$
Global Input Interrupt Status (GIIS)	GIIS00	CJP	TRUE	RTN00	0/B	$R \vee S$	0	Y=F	-	CISR	-	-	$\overline{\text{TACK}}$	$\overline{\text{ALUOE}}$, $\overline{\text{IBDRLE}}$, $\overline{\text{SCDG}}$
Global Activate Output (GAO)	GA000	CJP	OE	GA002	-	-	0	-	-	-	-	-	$\overline{\text{TACK}}$	$\overline{\text{SCDG}}$
	GA001	CJP	TRUE	RTN00	-	-	0	-	-	-	-	-	-	-
	GA002	CONT	-	-	D/A	$R \vee S$	0	Y=F F→B	CISR	CISR	OAFW	-	-	$\overline{\text{PR90E}}$, $\overline{\text{PROMCS}}$
	GA003	CJS	TRUE	DMAR00	D/0	$R \vee S$	0	Y=F F→B	-	OAR	-	-	-	$\overline{\text{IBDROE}}$, $\overline{\text{ALUOE}}$, $\overline{\text{IBARLE}}$
	GA004	CJP	GHLVM	GA007	D/0	$R \vee S$	0	Y=F F→B	-	OLR	-	-	-	$\overline{\text{IBDROE}}$
	GA005	CONT	-	-	D/A	R - S	0	Y=F	OLR	-	8Con	-	-	$\overline{\text{PR90E}}$, $\overline{\text{PROMCS}}$
	GA006	CJP	NEG	GA022	-	-	0	-	-	-	-	-	-	-
	GA007	CJS	TRUE	DMAR00	0/B	R + S	1	Y=F F→B	-	OAR	-	-	-	$\overline{\text{ALUOE}}$, $\overline{\text{IBARLE}}$
	GA008	CONT	-	-	D/A	$R \vee S$	0	Y=F F→B	CISR	CISR	OPFW	-	OP	$\overline{\text{IBDROE}}$, ODBPL, $\overline{\text{SCDG}}$ $\overline{\text{PR90E}}$, $\overline{\text{PROMCS}}$
	GA009	CONT	-	-	0/B	S - R	0	Y=F F→B	-	OLR	-	-	-	-
	GA010	CJP	ZERO	GA017	-	-	0	-	-	-	-	-	-	-
	GA011	CJP	OP	GA013	-	-	0	-	-	-	-	-	-	-

TABLE C-IV

(continued)

Microroutine	MP Addr	Next Addr Sel Instr	Cond Code Mux Sel	Branch Addr	ALU Source (R/S)	ALU Function	CN	ALU Dest Control	ALU A Addr	ALU B Addr	PROM Addr/ STK Instr	ICU Instr	Set/ Clear	Control
Global Activate Output (GAO) (Cont'd)	GA012	CONT	-	-	D/A	R ∧ S	0	Y=F F→B	CISR	CISR	$\overline{OPFW}$	-	-	$\overline{PR9OE}$, $\overline{PROMCS}$
	GA013	CJS	INTR	S100	-	-	0	-	-	-	-	-	-	-
	GA014	CJP	OE	GA017	-	-	0	-	-	-	-	-	-	-
	GA015	CONT	-	-	D/A	R ∧ S	0	Y=F F→B	CISR	CISR	$\overline{OAFW}$	-	-	$\overline{PR9OE}$, $\overline{PROMCS}$
	GA016	CJP	TRUE	RTN00	D/A	R ∧ S	0	Y=F F→B	CISR	CISR	$\overline{OPFW}$	-	$\overline{OPCL}$	$\overline{PR9OE}$, $\overline{PROMCS}$, $\overline{SCDG}$
	GA017	CJS	TRUE	DMAR00	O/B	R + S	1	Y=F F→B	-	OAR	-	-	-	$\overline{ALUOE}$, $\overline{IBARLE}$
	GA018	CJP	TRUE	GA010	O/B	S - R	0	Y=F F→B	-	OLR	-	-	-	-
	GA019	CONT	-	-	D/A	R ∧ S	0	Y=F F→B	CISR	CISR	$\overline{OAFW}$	-	-	$\overline{PR9OE}$, $\overline{PROMCS}$
	GA020	CJP	EMS(T)	GTC00	-	-	0	-	-	-	-	-	-	-
	GA021	CJP	TRUE	RTN00	-	-	0	-	-	-	-	-	GTCI	$\overline{SCDG}$
	GA022	CJP	TRUE	GMLV00	D/A	R ∧ S	0	Y=F F→B	CISR	CISR	$\overline{OAFW}$	-	-	$\overline{PR9OE}$, $\overline{PROMCS}$
Global Output Bus Control (GOBC)	GOBC00	CONT	-	-	-	-	0	-	-	-	-	-	$\overline{TACK}$	$\overline{IBDROE}$, $\overline{BLRG}$. $\overline{SCDG}$
	GOBC01	CJP	TRUE	RTN00	-	-	0	-	-	-	-	-	-	$\overline{BPRIOE}$, $\overline{BPRLD}$
Global Input Present Position (GIPP)	GIPP00	CJP	TRUE	RTN00	-	-	0	-	-	-	-	-	$\overline{TACK}$	$\overline{BPROE}$. $\overline{IBDRLE}$,$\overline{SCDG}$
Global Input Queue Pointer (GIQP)	GIQP00	CJP	TRUE	RTN00	-	-	0	-	-	-	-	-	$\overline{TACK}$	$\overline{IQPROE}$, $\overline{IBDRLE}$, $\overline{SCDG}$
Global Enable Output (GEO)	GE000	CONT	-	-	-	-	0	-	-	-	-	-	$\overline{TACK}$	$\overline{SCDG}$
	GE001	CJP	TRUE	RTN00	D/A	R ∨ S	0	Y=F F→B	CISR	CISR	OECon	-	OE	$\overline{SCDG}$, $\overline{PR9OE}$, $\overline{PROMCS}$

TABLE C-IV

(continued)

Microroutine	MP Addr	Next Addr Sel Instr	Cond Code Mux Sel	Branch Addr	ALU Source (R/S)	ALU Function	CN	ALU Dest Control	ALU A Addr	ALU B Addr	PROM Addr/ STK Instr	ICU Instr	Set/ Clear	Control
Global Disable Output (GDO)	GD000	CONT	-	-	-	-	0	-	-	-	-	-	$\overline{\text{TACK}}$	$\overline{\text{SCDG}}$
	GD001	CJP	TRUE	RTN00	D/A	R v S	0	Y=F F→B	CISR	CISR	$\overline{\text{OECon}}$	-	$\overline{\text{OECL}}$	$\overline{\text{SCDG}}$, $\overline{\text{PR9OE}}$, $\overline{\text{PROMCS}}$
Switch to Alternate Bus (STAB)	STAB00	CONT	-	-	-	-	0	-	-	-	-	-	$\overline{\text{TACK}}$	$\overline{\text{SCDG}}$
	STAB01	CJP	TRUE	RTN00	-	-	0	-	-	-	-	-	$\overline{\text{STAB}}$	$\overline{\text{SCDG}}$

Appendix D

Microword Fields

The tables for the various microword fields presented in Chapter IV are compiled in this appendix for the convenience of the reader.

TABLE D-I

Next Address Select Instruction Field

Microcode (Decimal) $I_0 - I_3$	Mnemonic	Name	Reg/Cntr Contents	$\overline{CCEN}$ = LOW and $\overline{CC}$ = HIGH		$\overline{CCEN}$ = HIGH or $\overline{CC}$ = LOW		Reg/ Cntr	Enable
				Y	Stack	Y	Stack		
0	JZ	Jump Zero	X	0	Clear	0	Clear	Hold	PL
1	CJS	Cond JSB PL	X	PC	Hold	D	Push	Hold	PL
2	JMAP	Jump Map	X	D	Hold	D	Hold	Hold	MAP
3	CJP	Cond Jump PL	X	PC	Hold	D	Hold	Hold	PL
4	PUSH	Push/Cond LD Cntr	X	PC	Push	PC	Push	Note 1	PL
5	JSRP	Cond JSB R/PL	X	R	Push	D	Push	Hold	PL
6	CJV	Cond Jump Vector	X	PC	Hold	D	Hold	Hold	VECT
7	JRP	Cond Jump R/PL	X	R	Hold	D	Hold	Hold	PL
8	RFCT	Repeat Loop, Cntr ≠ 0	≠0	F	Hold	F	Hold	Dec	PL
			=0	PC	Pop	PC	Pop	Hold	PL
9	RPCT	Repeat PL, Cntr ≠ 0	≠0	D	Hold	D	Hold	Dec	PL
			=0	PC	Hold	PC	Hold	Hold	PL
10	CRTN	Cond RTN	X	PC	Hold	F	Pop	Hold	PL
11	CJPP	Cond Jump PL & Pop	X	PC	Hold	D	Pop	Hold	PL
12	LDCT	LD Cntr & Continue	X	PC	Hold	PC	Hold	LOAD	PL
13	LOOP	Test End Loop	X	F	Hold	PC	Pop	Hold	PL
14	CONT	Continue	X	PC	Hold	PC	Hold	Hold	PL
15	TWB	Three-way Branch	≠0	F	Hold	PC	Pop	Dec	PL
			=0	D	Pop	PC	Pop	Hold	PL

NOTE 1:. If $\overline{CCEN}$ = LOW and $\overline{CC}$ = HIGH, Hold; else load. X = Don't Care

TABLE D-II

Condition Code Multiplexer Select Field

Micro Code (Decimal) I_4 -I_8	Mnemonic	Description
0	TRUE	True Signal
1	INTR	Interrupt request
2	NEG	ALU output negative
3	ZERO	ALU output zero
4	$\overline{\text{ZERO}}$	ALU output not zero
5	$\overline{\text{BGI}}$	Bus Grant (In) signal low
6	TACK	Transfer Acknowledge signal high
7	$\overline{\text{DTACK}}$	Delayed Transfer Acknowledge signal low
8	IAK	Interrupt Acknowledge signal high
9	INHB	Inhibit signal high
10	CIVBOR	CPU Interrupt Vector Buffer output ready
11	MHS	Message Header Sync received
12	EMS(R)	End Message Sync received
13	EMS(T)	End Message Sync transmitted
14	GMHE	Global message header error
15	OP	Output pending
16	RB	Response bit set
17	GHP	Global high priority message
18	GBDD	Global bus dominance disabled
19	OE	Output enabled
20	GHPMRM	GHPMR interrupt masked
21	GMLVM	GMLV interrupt masked
22	GMHEM	GMHE interrupt masked
23	GMWCEM	GMWCE interrupt masked
24	GTTOM	GTTO interrupt masked
25	GMRM	GMR interrupt masked
26	GTCM	GTC interrupt masked

TABLE D-III

ALU Source Operand Field

Microcode (Decimal) I_{17}-I_{19}	ALU Source Operands R	S
0	A	Q
1	A	B
2	0	Q
3	0	B
4	0	A
5	D	A
6	D	Q
7	D	0

Notes: "A" designates RAM A-port
"B" designates RAM B-port
"Q" designates Q Register
"D" designates direct data input

TABLE D-IV
ALU Function Field

Microcode (Decimal) I_{20}-I_{22}	ALU Function	Symbol
0	R Plus S	$R + S$
1	S Minus R	$S - R$
2	R Minus S	$R - S$
3	R OR S	$R \vee S$
4	R AND S	$R \wedge S$
5	$\overline{R}$ AND S	$\overline{R} \wedge S$
6	R EX-OR S	$R \forall S$
7	R EX-NOR S	$\overline{R \forall S}$

TABLE D-V

ALU Destination Control Field

Microcode (Decimal)	RAM Function		Q-Reg. Function		Output
I_{24} - I_{26}	Shift	Load	Shift	Load	Y
0	X	None	None	F-Q	F
1	X	None	X	None	F
2	None	F-B	X	None	A
3	None	F-B	X	None	F
4	Down	F/2-B	Down	Q/2-Q	F
5	Down	F/2-B	X	None	F
6	Up	2F-B	Up	2Q-Q	F
7	Up	2F-B	X	None	F

Notes: "X" designates Don't care
"F" designates internal ALU output
"Up" is towards most significant bit
"Down" is towards least significant bit

TABLE D-VI

PROM Addr/Stack Instruction Field

PROM Addr Mnemonic	Value (Hexidecimal)	Function
OCon	0000	Constant, 0_{10}
8Con	0008	Constant, 8_{10}
WIBF	0007	Word used for Direct Memory Access Write operation. Three least significant bits contain complement of $\overline{\text{TRQ}}$, $\overline{\text{IOSL}}$, & $\overline{\text{DRCV}}$.
RIBF	0006	Word used for Direct Memory Read Operation. Three least significant bits contain complement of $\overline{\text{TRQ}}$, $\overline{\text{IOSL}}$, & DRCV.
$\overline{\text{BDM}}$	D777	Constant used to clear GBD mask
MIDZC	003F	Constant used to clear ten most significant bits
MIDC	To be deter.	Constant used to obtain MIAMM address. Seventh and eighth least significant bits contain value, other bits are zero.
MBZC	03FF	Constant used to clear six most significant bits
MBC	To be deter.	Constant used to obtain Message Buffer address. Six most significant bits contain value, other bits are zero.
BDM	2000	Constant used to set GBD mask
GBQIV	To be deter	GBO Interrupt vector
GMWLEIV		GMWLE interrupt vector
GMPEIV		GMPE interrupt vector
GMEEIV		GMEE interrupt vector
GTTOIV		GTTO interrupt vector
GHPMRIV		GHPMR interrupt vector
GMLVIV		GMLV interrupt vector

TABLE D-VI
(Continued)

PROM Addr Mnemonic	Value (Hexidecimal)	Function
GMHEIV	To be deter.	GMHE interrupt vector
GMWCEIV	↓	GMWCE interrupt vector
GMRIV	↓	GMR interrupt vector
GTCIV	↓	GTC interrupt vector
IM	7FF0	Constant used to process interrupt mask
OAFW	0002	Constant used to set Output Active Flag
$\overline{\text{OAFW}}$	FFFD	Constant used to clear Output Active Flag
OPFW	0004	Constant used to set Output Pending Flag
$\overline{\text{OPFW}}$	FFFB	Constant used to clear Output Pending Flag
OECon	0001	Constant used to set Output Enable Flag
$\overline{\text{OECon}}$	FFFE	Constant used to clear Output Enable Flag
MIAMM00	To be deter.	MIAMM addresses
MIAMM63	↓	↓
STK Instr		
PD	000C	Push D, push "D" input onto stack
PS	000D	Pop S, pop stack

TABLE D-VII

Interrupt Control Unit Instruction Field

Microcode (Decimal) I_{43} -I_{46}	IE	Function
0	0	Master Clear
1	0	Clear All Interrupts
2	0	Clear Interrupts via Y-Bus
3	0	Clear Interrupt via Mask Register
4	0	Clear Interrupt, Last Vector Read
5	0	Read Vector
6	0	Read Status Register
7	0	Read Mask Register
8	0	Set Mask Register
9	0	Load Status Register
10	0	Bit Clear Mask Register
11	0	Bit Set Mask Register
12	0	Clear Mask Register
13	0	Disable Request
14	0	Load Mask Register
15	0	Enable Request
X	1	Instruction Disable

Note: "X" designates Don't Care

TABLE D-VIII

Set/Clear Field Signals

Mnemonic	Name
$\overline{\text{WRCL}}$	Word Received Interrupt Clear
$\overline{\text{GMWLECL}}$	Global Message Word Length Error Intr Clr
$\overline{\text{GMPECL}}$	Global Message Parity Error Intr Clear
$\overline{\text{GMEECL}}$	Global Message Encoding Error Intr Clear
$\overline{\text{STAB}}$	Switch to Alternate Bus
$\overline{\text{MHSCL}}$	Message Header Sync Clear
$\overline{\text{EMS(R)CL}}$	End Message Sync (Received) Clear
$\overline{\text{EMS(T)CL}}$	End Message Sync (Transmitted) Clear
GMHE	Global Message Header Error
OP	Output Pending
$\overline{\text{OPCL}}$	Output Pending Clear
RB	Response Bit
$\overline{\text{RBCL}}$	Response Bit Clear
GHP	Global High Priority
$\overline{\text{GHPCL}}$	Global High Priority Clear
GBDD	Global Bus Dominance Disabled
$\overline{\text{GBDDCL}}$	Global Bus Dominance Disabled Clear
OE	Output Enable
$\overline{\text{OECL}}$	Output Enable Clear
$\overline{\text{GHPMRI}}$	Global High Priority Message Received Intr
$\overline{\text{GMLVI}}$	Global Message Length Violation Interrupt
$\overline{\text{GMHEI}}$	Global Message Header Error Interrupt

TABLE D-VIII

(Continued)

Mneomonic	**Name**
$\overline{\text{GMWCEI}}$	Global Message Word Count Error Interrupt
$\overline{\text{GTCI}}$	Global Transmission Completed Interrupt
GTCI	Global Transmission Completed Interrupt
$\overline{\text{GTCCL}}$	Global Transmission Completed Clear
BRQ	Bus Request
$\overline{\text{TACK}}$	Transfer Acknowledge
$\overline{\text{IRQ}}$	Interrupt Request
$\overline{\text{IRQCL}}$	Interrupt Request Clear
$\overline{\text{ITACK}}$	Interrupt Transfer Acknowledge
$\overline{\text{ITACKCL}}$	Interrupt Transfer Acknowledge Clear

TABLE D-IX

Control Field Signals

Mnemonic	Name
$\overline{\mathrm{BTUOE}}$	Bus Translation Unit Output Enable
$\overline{\mathrm{BLRG}}$	Bus Length Register Enable
$\overline{\mathrm{BPRIOE}}$	Bus Position Register (In) Output Enable
$\overline{\mathrm{BPRLD}}$	Bus Position Register Load
$\overline{\mathrm{BPROE}}$	Bus Position Register Output Enable
ODBPL	Output Data Buffer Parallel Load
$\overline{\mathrm{SCDG}}$	Set/Clear Decoder Enable
$\overline{\mathrm{ICUIE}}$	Interrupt Control Unit Instruction Enable
$\overline{\mathrm{CCLG}}$	Condition Code Latch Enable
$\overline{\mathrm{MCRLD}}$	Microprogram Controller Register Load
MCCI	Microprogram Controller Carry-In
$\overline{\mathrm{PR9OE}}$	Pipeline Register 9 Output Enable
$\overline{\mathrm{PROMCS}}$	BIU PROM Select
PARLE	PROM Address Register Later Enable
$\overline{\mathrm{PAROE}}$	PROM Address Register Output Enable
$\overline{\mathrm{ALUOE}}$	ALU Output Enable
$\overline{\mathrm{RBRG}}$	Response Bit Register Enable
$\overline{\mathrm{RBRDOE}}$	Response Bit Register Decoder Output Enable
$\overline{\mathrm{STKIEN}}$	BIU Stack Instruction Enable
$\overline{\mathrm{STKOE}}$	BIU Stack Output Enable
CIVBPL	CPU Interrupt Vector Buffer Parallel Load

TABLE D-IX
(Continued)

Mnemonic	Name
CIVBPD	CPU Interrupt Vector Parallel Dump
$\overline{\text{IQPRCP}}$	Input Queue Pointer Register Clock Pulse
$\overline{\text{IQPROE}}$	Input Queue Pointer Register Output Enable
$\overline{\text{IBFRLE}}$	I-Bus Flag Register Latch Enable
$\overline{\text{IBFRBE}}$	I-Bus Flag Register Bus Enable
$\overline{\text{IBFRCL}}$	I-Bus Flag Register Flag Clear
$\overline{\text{IBARLE}}$	I-Bus Address Register Latch Enable
$\overline{\text{IBARBE}}$	I-Bus Address Register Bus Enable
$\overline{\text{IBARCL}}$	I-Bus Address Register Flag Clear
$\overline{\text{IBARRLE}}$	I-Bus Address Register Rcvr Latch Enable
$\overline{\text{IBAROE}}$	I-Bus Address Register Output Enable
$\overline{\text{IBDRLE}}$	I-Bus Data Register Latch Enable
$\overline{\text{IBDRBE}}$	I-Bus Data Register Bus Enable
$\overline{\text{IBDRCL}}$	I-Bus Data Register Flag Clear
$\overline{\text{IBDRRLE}}$	I-Bus Data Register Receiver Latch Enable
$\overline{\text{IBDROE}}$	I-Bus Data Register Output Enable

NOTE: Three bits of the field are not used.

VITA

Leslie T. Konno was born on 28 December 1946 in Kahului, Maui, Hawaii. In June 1964, he graduated from Kaimuki High School in Honolulu, Hawaii. He attended the University of Hawaii and received a Bachelor of Science Degree in Electrical Engineering in January 1969. Through the Air Force Reserve Officer Training Corps, he was commissioned a Second Lieutenant in the United States Air Force in February 1969. From May 1969 through December 1974, he was assigned to the Tactical Loran SPO, first at the Aeronautical Systems Division, Wright-Patterson Air Force Base, Ohio and then at the Electronic Systems Division, Hanscom Field, Massachusetts. He spent the next two years at the Kaena Point Satellite Tracking Station in Hawaii. He entered the Air Force Institute of Technology resident school in June 1976 and received the degree of Master of Science in Electrical Engineering in December 1977.

Permanent Address: 1870 Paula Drive
Honolulu, Hawaii 96816

14 **REPORT DOCUMENTATION PAGE** 9	READ INSTRUCTIONS BEFORE COMPLETING FORM
~~1. REPORT NUMBER~~ AFIT/GE/EE/77-24	2. GOVT ACCESSION NO. ~~3. RECIPIENT'S CATALOG NUMBER~~ Master's thesis,
~~4. TITLE (and Subtitle)~~ 6 BUS INTERFACE UNIT DESIGN FOR THE DISTRIBUTED PROCESSOR/MEMORY SYSTEM .	~~5. TYPE OF REPORT & PERIOD COVERED~~ M.S. Thesis
	6. PERFORMING ORG. REPORT NUMBER
~~7. AUTHOR(s)~~ 10 Leslie T. Konno Captain USAF	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB, OH 45433	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Avionics Laboratory (AFAL/AAT) Wright-Patterson AFB, OH 45433	11 ~~12. REPORT DATE~~ Dec[redacted]977 12
	~~13. NUMBER OF PAGES~~ 232 237 p.
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*	15. SECURITY CLASS. *(of ~~this report~~)* UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

Approved for public release: IAW AFR 190-17

Jerral F. Guess
JERRAL F. GUESS, Captain, USAF
Director of Information

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Emulator	Bus Interface
Microprocessor	Avionics
Bit-Slice Microprocessor	
Interrupt Driven Microprocessor	

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This report describes the design of the Bus Interface Unit for the Distributed Processor/Memory (DP/M) System. The DP/M System, which is being developed by the Air Force Avionics Laboratory, is a concept to integrate avionics on board an aircraft by utilizing a number of programmable processor/memory elements (PE's) in a distributed (decentralized) network. All the PE's in the system are interconnected by a pair of redundant global buses and PE's in an affinity group are additionally interconnected by a local bus.

This effort involves the design of the Bus Interface Unit (BIU) of the PE.

alt

The BIU interfaces the parallel-data PE processor with the two redundant serial global buses and the serial local bus. The BIU has been designed as an interrupt driven microprogrammed processor. The design uses a bipolar bit-slice microprocessor, specifically the Am2900 Bipolar Microprocessor Family, for emulating the BIU functions.

This report starts with a brief description of the DP/M System, followed by a detailed description of the BIU functions. Then the hard-ware, the microword format, and the microroutines are described. A discussion on the design effort is presented at the end, and recommendations are made for system improvement.